Volume 10 Number 3

ACTA CYBERNETICA

Szeged, 1992

Information for authors: Acta Cybernetica publishes only original papers in English in the field of computer sciences. Review papers are accepted only exceptionally. Manuscripts should be sent in triplicate to one of the Editors. The manuscript must be typed double-spaced on one side of the paper only. For the form of references, see one of the articles previously published in the journal.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

On the randomized complexity of monotone graph properties

Gröger Hans Dietmar *

1 Introduction

Let $C^R(P)$ be the number of questions of the form 'Does the graph G contain the edge $e(i,j)$?' that have to be asked in the worst case by any randomized decision tree algorithm for computing an n-vertex graph property P. For non-trivial, monotone graph properties it is known, that the deterministic complexity is $\Omega(n^2)$ (see [4]). R. Karp [5] conjectured, that this bound holds for randomized algorithms as well. As far as this conjecture we know the following results. The best uniform lower bound for all non-trivial, monotone graph properties is $\Omega(n^{4/3})$ due to P. Hajnal [1].

No non-trivial, monotone graph property is known having a randomized complexity of less than $n^2/4$. Some properties have been proven to have complexity of $\Omega(n^2)$ (see A. Yao [6]).

In this paper we refine the idea of Yao. This leads to a further improvement in the reductions of arbitrary graph properties to bipartite graph properties. (see [1], [3]) and yields a uniform lower bound for the subgraph isomorphism properties of $\Omega(n^{3/2})$. Furthermore we show, that a large variety of isomorphism properties as well as k-colourability require $\Omega(n^2)$ questions.

2 Preliminaries, notations

A decision tree is a rooted binary tree with labels on each node and edge. Each inner node is labeled by a variable symbol and the two edges leaving the node are labeled by 0 and 1. Each leaf is also labeled by 0 or 1. Obviously, any truth-assignment of the variables determines a unique path from the root to a leaf.

A decision tree A *computes a boolean function* f if for all input $\underline{x}$ the corresponding path in A leads to a leaf labeled by $f(\underline{x})$.

Let $cost(A,\underline{x})$ be the number of questions asked when the decision tree A is executed on input $\underline{x}$. This is the length of the path induced by $\underline{x}$. The *deterministic decision tree complexity* of a boolean function f is $C(f) = \min_A \max_{\underline{x}} cost(A,\underline{x})$, where the minimum is taken over all decision trees A computing the function f.

*Department of Applied Computer Sciences József Attila University 6720 Szeged, Árpád tér 2.

In a *randomized decision tree* the question asked next not only depends on the answers it got so far but also on the outcome of a trial. Since all trials can be done in advance we can view a randomised decision tree as a probability distribution on the set of deterministic trees. *A randomized decision tree computes a boolean function f* iff the distribution is non-zero only on deterministic trees computing f.

Definition 2.1 *Let $\{A_1, \ldots, A_N\}$ be the set of all deterministic decision trees computing f. Let $R = \{p_1, \ldots, p_N\}$ be a randomized decision tree, where p_i denotes the probability of A_i. The* cost of R on input $\underline{x}$ *is* $cost(R, x) = \sum_i p_i \cdot cost(A_i, \underline{x})$. *The* randomized decision tree complexity *of a function f is*

$$C^R(f) = \min_R \max_{\underline{x}} cost(R, \underline{x}),$$

where the minimum is taken over all randomized decision trees computing the function f. The following lemma yields the base of all lower bound proofs for randomized decision tree complexity so far.

Lemma 2.2 *(A. Yao [6]) Let d be a probability distribution on the set of all possible inputs and let $d(\underline{x})$ be the probability of input $\underline{x}$. We define the* average case *performance of a deterministic tree A computing f as* $av(A, d) = \sum_{\underline{x}} d(\underline{x}) cost(A, \underline{x})$.

Then for any boolean function f

$$C^R(f) = \max_d \min_A av(A, d),$$

where the minimum is taken over all deterministic decision trees computing f.

A boolean function f is called *non-trivial, monotone* iff $f(\underline{0}) = f(\underline{1}) = 1$ and $f(\underline{x}_1) \leq f(\underline{x}_2)$ for all $\underline{x}_1 \leq \underline{x}_2$. Here we mean component wise less or equal. In this paper we deal only with graph properties and bipartite graph properties. Since a graph on n vertices can be identified with a $(0,1)$-string of length $\binom{n}{2}$, a graph property can be given by a boolean function which takes equal values on isomorphic graphs. So, by *graph property* we mean a suitable boolean function and sometimes instead of the function we give the property by the set of all graphs having this property. *A* graph property is called *non-trivial, monotone* iff the corresponding boolean function is non-trivial, monotone.

Let us denote the set of all n-vertex by $\mathcal{G}_n$ and the set of all non-trivial, monotone graph properties defined on $\mathcal{G}_n$ by $\mathcal{P}_n$. Clearly, a property $P \in \mathcal{P}_n$ can be characterized by the set of minimal graphs having that property. Let $\min(P)$ be the list of minimal graphs for P. If $\min(P)$ contains up to isomorphism only one graph G, we call P a *subgraph isomorphism property* and denote it by P_G.

Let us denote by $d_G(x)$ the degree of a node x in G, by $D(G)$ the maximal degree of G, by $\delta(G)$ the minimal degree of G and by $\overline{d}(G)$ the average degree of G. Furthermore, denote $V(G)$ the set of vertices with non-zero degree of G, $E(G)$ the set of edges of G and K_n, E_n the complete and the empty graph on n nodes, respectively. Sometimes we use the disjoint union of K_{n-r} and E_r, and this graph is denoted by K^*_{n-r}.

Let $0 < m < n$ and $P \in \mathcal{P}_n$. Using the property P, we can define two (not necessarily non-trivial) monotone graph properties on $\mathcal{G}_m$. For this reason, divide the set of nodes, P is defined on, into disjoint sets V_1 and V_2 so that $|V_1| = m$ and $|V_2| = n - m$. Let $ind(P|m)$ and $red(P|m)$ denote the following m-vertex properties:

$G \in ind(P|m)$ iff adding all nodes in V_2 to G and keeping the original edge-set, we obtain a graph having property P.

$G \in red(P|m)$ iff adding all nodes in V_2 together with all possible edges incident to them to G, we get a graph having property P.

Obviously $C^R(ind(P|m)) \leq C^R(P)$ and $C^R(red(P|m)) \leq C^R(P)$.

We have to build up the same system of notions for the universe of labeled bipartite graphs with colour classes $V = \{1, 2, \ldots, n\}$ and $W = \{\overline{1}, \overline{2}, \ldots, \overline{m}\}$ denoted by $\mathcal{G}_{n,m}$. The set of all non-trivial, monotone bipartite graph properties on $\mathcal{G}_{n,m}$ is denoted by $\mathcal{P}_{n,m}$. We also use the other corresponding notions $C^R(P)$, $\min(P)$ and $E(G)$.

If $G \in \mathcal{G}_{n,m}$ and U is a subset of the vertices then let us denote by $d_{\max,U}(G)$ and $d_{av,U}(G)$ the maximal and average degree in the set U, and by $K_{n,m}$, $E_{n,m}$ the complete bipartite graph and the empty bipartite graph, respectively

Let $0 < r < n$ and $P \in \mathcal{P}_{n,m}$. Divide V into disjoint sets V_1 and V_2 so that $|V_1| = r$ and $|V_2| = n - r$. Let $ind_V(P|r)$ and $red_V(P|r)$ denote the following bipartite graph properties defined on $\mathcal{G}_{n,m}$:

$G \in ind_V(P|r)$ iff adding all nodes of V_2 to G, we obtain a bipartite graph having property P.

$G \in red_V(P|r)$ iff adding all nodes of V_2 together with all possible edges between V_2 and W to G, we get a bipartite graph having property P.

Obviously $C^R(ind_V(P|r)) \leq C^R(P)$ and $C^R(red_V(P|r)) \leq C^R(P)$.

Finally let

$$C^R(n, m) = \min\{C^R(P) | P \in \mathcal{P}_{n,m}\}.$$

In lower bound proofs for the complexity of monotone graph properties the following reduction to bipartite graph properties plays an important role.

Let $P \in \mathcal{P}_n$ and $0 < r < n$. Furthermore, let $bipart(P|r, n-r)$ be the following bipartite graph property defined on $\mathcal{G}_{r,n-r}$

$G \in bipart(P|r, n-r)$ iff adding all edges between nodes in W, we obtain a graph having property P.

Obviously $C^R(bipart(P|r, n-r)) \leq C^R(P)$ and so if $bipart(P|r, n-r)$ is non-trivial, then $C^R(r, n-r) \leq C^R(P)$.

A good survey of previous techniques can be found in [1]. We only mention those, we will apply.

Theorem 2.3 *(Basic Method [6]) (i) Let $P \in \mathcal{P}_n$ and $G \in \min(P)$ be any minimal graph for P. Then*

$$C^R(P) \geq |E(G)|.$$

(ii) Let $P \in \mathcal{P}_{n,m}$ and $G \in \min(P)$ be any minimal graph for P. Then

$$C^R(P) \geq |E(G)|.$$

Definition 2.4 *Let $\mathcal{L}$ be a list of graphs from $\mathcal{G}_{n,m}$. For each $G \in \mathcal{L}$ let us consider the sequence of degrees in colour clas V. Let $d_1 \geq d_2 \geq \ldots \geq d_n$ be the ordered list of degrees. If $(d_1, d_2, \ldots, d_n)$ is the lexicographically minimal sequence considering all the ordered lists then we refer to G as the V-*lexicographically first element *of $\mathcal{L}$*

Theorem 2.5 *(Yao's Method [7]) Let $P \in \mathcal{P}_{n,m}$ and G be the V-lexicographically first graph of $\min(P)$. Then*

$$C^R(P) = \Omega\left(\frac{d_{\max,V}(G)}{d_{av,V}(G)} \cdot |V|\right).$$

A very useful tool for proving lower bounds is dualiy. For every non-trivial, monotone boolean function f we can define the dual function f^D *as follows:*

$$f^D(\underline{x}) = \neg f(\neg \underline{x}).$$

It is easy to see that f^D is also non-trivial, monotone and $C^R(f^D) = C^R(f)$.

Definition 2.6 *(i) Let $G, H \in \mathcal{G}_n$ with vertex sets V and V', respectively. A* packing *is an identification between V and V' such that no edge of G is identified with any edge of H.*

(ii) Let $G, H \in \mathcal{G}_{n,m}$ with colour classes V, W and V', W', respectively. A bipartite packing *is an identification between V and V' and between W and W' such that no edge of G is identified with any edge of H.*

Lemma 2.7 *(Yao [6]) (i) If $P \in \mathcal{P}_n, G \in \min(P)$ and $H \in \min(P^D)$ then G and H can't be packed. (ii) If $P \in \mathcal{P}_{n,m}, G \in \min(P)$ and $H \in \min(P^D)$ then G and H can't be packed as bipartite graphs.*

3 Results

By a *covering* of a graph G we mean a subset K of V such that any edge of G is adjacent to at least one vertex in K. A covering K is *minimal* if G has no covering K' with $|K'| < |K|$.

The *width* of a graph G denoted by $\mathrm{width}(G)$ is the size of a minimal covering of G. The *trace* of a graph G denoted by $\mathrm{trace}(G)$ is the minimal number of edges we have to remove from G in order to decrease its width.

Now we extend these notions to monotone graph properties. The *width of a monotone graph property* P is defined as follows:

$$width(P) = \min\{width(G) | G \in \min(P)\}$$

The *trace of a monotone graph property* P is defined by

$$trace(P) = \min\{trace(G) | G \in \min(P) \text{ and } width(P) = width(G)\}$$

The following assertions show some fundamental properties of these notions.

Lemma 3.1 *If $P \in \mathcal{P}_n$ and $1 \leq r < n$ is a fixed integer then*

(i) $width(P) \geq r$ iff $K^*_{n+1-r} \in P^D$

(ii) If $width(P) > r$ then

$$red(P|n-r) \in \mathcal{P}_{n-r}, width(red(P|n-r)) = width(P) - r, trace(red(P|n-r)) = trace(P).$$

Lemma 3.2 *If $P \in cal P_n$ and $width(P) = 1$ then for any $G \in P^D, G$ has at least $\frac{1}{2}n \cdot (n - trace(P))$ edges.*

Proof. Since P^D is a non-trivial, monoton graph property, it is sufficient to prove the statement for $G \in \min(P^D)$. Indeed, let $G \in \min(P^D)$ be arbitrary and let $H \in \min(P)$ such a graph for which $\mathrm{width}(H) = \mathrm{width}(P)$ and $\mathrm{trace}(H) = \mathrm{trace}(P)$ holds. With other words, H is a star with $\mathrm{trace}(P)$ mary edges. According to Lemma 2.7. G and H can't be packed. This implies $\delta(G) \geq |V(G)| - trace(H) = n - trace(H)$, therefore $|E(G)| \geq \frac{1}{2}n \cdot (n - trace(H))$.

Lemma 3.3 *For any $P \in \mathcal{P}_n$ the following assertions hold:*

(i) $C^R(P) \geq width(P) \cdot trace(P)$.

(ii) $C^R(P) \geq \frac{1}{2}(n+1-width(P)) \cdot (n+1-(width(P)+trace(P)))$.

(iii) For any $0 < \varepsilon < 1$, *if* $width(P) \leq (1-\varepsilon) \cdot n$ *then* $C^R(P) \geq \frac{\varepsilon^3}{2-\varepsilon} n \cdot width(P)$.

Proof. Assertion (i) is a straightforward consequence of Theorem 2.3. To prove (ii) choose in Lemma 3.1. r =width $(P) - 1$ and apply Lemma 3.2. to the reduced property re $\alpha(P|n - (\text{width } (P) - 1))$. Finally Theorem 2.3. yields the result. If $\text{trace}(P) \geq \frac{\varepsilon^2}{2-\varepsilon} \cdot n$, then assertion (iii) follows from (i), else it can be proved, using (ii) and assumption $\text{width}(P) \leq (1-\varepsilon) \cdot n$.

Before we state our main results we apply this method to some special graph properties. For this reason let us denote the property that an n-vertex graph contains a Hamiltonian cycle by PH_n and the property that an n-vertex graph has a vertex colouring with k colours by $PC_{k,n}$.

Theorem 3.4

$$C^R(PH_n) \geq \frac{1}{8} \cdot (n^2 - 1)$$

$$C^R(PC_{n,k}) \geq \binom{n+1-k}{2}.$$

Proof. We have only to determine the width and trace of the given properties. The required values are:

$$width(PH_n) = \lceil \frac{n}{2} \rceil,$$

$$trace(PH_n) = \begin{cases} 1, & \text{if } n \text{ is odd} \\ 2, & \text{if } n \text{ is even} \end{cases}$$

Since $PC_{k,n}$ itself is not monotone, we consider instead of $PC_{k,n}$ the property $\neg PC_{k,n}$ which is non-trivial, monotone and obviously, $C^R(\neg PC_{k,n}) = C^R(PC_{k,n})$. It can be seen that the corresponding values are:

$$width(\neg PC_{k,n}) = k$$

$$trace(\neg PC_{k,n}) = 1.$$

The following theorem improves the known reductions of non-trivial, monotone graph properties to bipartite graph properties. Although King [3] has already stated a similar result, the new approach can help to prove better uniform lower bounds, since the reduction is to colour classes both of size $\Omega(n)$

Theorem 3.5 *The randomized decision tree complexity of any non-trivial, monotone graph property $P \in \mathcal{P}_n$ is*

$$C^R(P) \geq \min\{\frac{1}{40} \cdot n^{3/2}, C^R(\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)\}.$$

Proof. We have only to consider the case that the property P can't be reduced to a non-trivial bipartite graph property $bipart P|\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)$. This implies $K^*_{\lfloor \frac{n}{2} \rfloor} \in P$ or $K^*_{\lceil \frac{n}{2} \rceil} \in P^D$. Therefor, it remains only to prove, that for any $P \in \mathcal{P}_n$, if $K^*_{\lceil \frac{n}{2} \rceil} \in P$ then $C^R(P) \geq \frac{1}{40} \cdot n^{3/2}$ holds.

Let us suppose that we found a property $P \in \mathcal{P}_n$ with $K^*_{\lceil \frac{n}{2} \rceil} \in P$ and $C^R(P) < \frac{1}{40} \cdot n^{3/2}$. Let us construct the following sequence of induced graph properties

$$\{P_i | 0 \leq i \leq \lfloor \frac{1}{2} n^{1/2} \rfloor\}, P_i = ind(P|\lceil \frac{3}{4}n + \frac{1}{2} i \cdot n^{1/2} \rceil).$$

Since $K^*_{\lceil \frac{n}{2} \rceil} \in P$ and for any i P_i is an induced property of P on at least $\lceil \frac{3}{4} n \rceil$ vertices, P_i is non-trivial and

$$C^R(P_i) \leq C^R(P) < \frac{1}{40} n^{3/2} \tag{1}$$

$K^*_{\lceil \frac{n}{2} \rceil} \in P_0$ implies $\text{width}(P_0) \leq \lceil \frac{n}{2} \rceil - 1 \leq \frac{1}{2} n$. Assertion (iii) of Lemma 3.3. yields $C^R(P_0) \geq \frac{1}{20} n \cdot \text{width}(P_0)$. Hence

$$width(P_0) \leq \lfloor \frac{1}{2} n^{1/2} \rfloor. \tag{2}$$

Obviously $G \in P_i$ implies $G \in P_{i+1}$. Therefore

$$width(P_{i+1}) \leq width(P_i), i \geq 0. \tag{3}$$

Let us suppose, that for some $i \geq 0$ $\text{width}(P_{i+1}) = \text{width}(P_i)$ holds. Then $\text{trace}(P_{i+1}) \leq \text{trace}(P_i)$. Now Lemma 3.3. yields

$$\begin{aligned} C^R(P_{i+1}) &\geq \frac{1}{2}(\lceil \frac{3}{4}n + \frac{1}{2}(i+1) \cdot n^{1/2} \rceil + 1 - width(P_{i+1})) \cdot \\ &\quad (\lceil \frac{3}{4}n + \frac{1}{2}(i+1) \cdot n^{1/2} \rceil + 1 - (width(P_{i+1}) + trace(P_{i+1}))) \\ &\geq \frac{1}{2} \cdot (\frac{3}{4}n + \frac{1}{2} n^{1/2} - width(P_0)) \cdot \\ &\quad (\frac{3}{4}n + \frac{1}{2} i \cdot n^{1/2} + 1 - (width(P_i) + trace(P_i)) + \frac{1}{2} n^{1/2}) \\ &\geq \frac{3}{16} n^{3/2} > \frac{1}{40} n^{3/2}, \end{aligned}$$

which contradicts (1).

The sequence of positive integers $\{\text{width } (P_i) | 0 \leq i \leq \lfloor \frac{1}{2} n^{1/2} \rfloor\}$ therefore decreases strictly monotone, and so

$$width(P_0) \geq \lfloor \frac{1}{2} n^{1/2} \rfloor + 1,$$

which contradicts (2).

Since our assumption $C^R(P) < \frac{1}{40} n^{3/2}$ led to a contradiction we have completed the proof.

A straightforward consequence of the improved reduction is the following result.

Theorem 3.6 *For the randomized decision tree complexity of any subgraph isomorphism property $P_G \in \mathcal{P}_n$*

$$C^R(P_G) = \Omega(n^{3/2}).$$

Proof. According to Theorem 3.5., we have only to settle the case that bipart$(P_G|\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)$ is nontrivial. Depending on width(P_G) and trace(P_G) we shall distinguish three cases.

Case 1. Assume that width$(P_G) \geq \frac{1}{4}n$. Since bipart$(P_G|\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)$ is non-trivial, we get width$(P_G) \leq \frac{1}{2}n$ and assertion (iii) of Lemma 3.3. implies a lower bound of $\Omega(n^2)$.

Case 2. If width$(P_G) < \frac{1}{4}n$ and trace$(P_G) < \frac{2}{3}n$, then we can apply assertion (ii) of Lemma 2.5. and get also a lower bound of $\Omega(n^2)$.

Case 3. Suppose that width$(P_G) < \frac{1}{4}n$ and trace$(P_G) \geq \frac{2}{3}n$. Since trace$(P_G) \geq \frac{2}{3}n$ the corresponding bipartite graph property bipart$(P_G|\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)$ has only such minimal graphs H for that $D_V(H) \geq \frac{1}{6}n$ holds. If bipart$(P_G|\lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil)$ has a minimal graph with at least $n^{3/2}$ edges we can apply Theorem 2.3. Otherwise we can apply Theorem 2.5. In both cases we get a lower bound of $\Omega(n^{3//2})$ which completes the proof.

Before we prove the sharper version of Theorem 2.6. we consider some special bipartite graph properties. Let us denote by $S_{n,m}$ the graph which has one vertex of positive degree in V and m edges.

Lemma 3.7 *Let $P_S \in \mathcal{P}_{n,m}$ denote the property of containing a subgraph isomorph to $S_{n,m}$. Then*

$$C^R(P_S) \geq \frac{1}{2}m \cdot n.$$

Proof. (analogue to Yao [7]). We consider the dual property P_S^D, which is easy to see to contain exactly those graphs, that have no isolated nodes in colour class W. According to Lemma 2.2. wee choose as a "hard" input distribution the uniform distribution over all minimal graphs. Let be A an optimal deterministic decision tree, that computes our P_S^D. We denote by $X_i(G)$ the number of edges incident to w_i asked by A. Then

$$\begin{aligned} C^R(P_S^D) &\geq E(\sum_{i=1}^{m} X_i(G)) \\ &= \sum_{i=1}^{m} E(X_i(G)) \end{aligned}$$

Since for any value of i we have to find one edge out of n edges, we get

$$E(X_i(G)) \geq \frac{1}{2}n$$

and finally

$$C^R(P_S) \geq \frac{1}{2}m \cdot n.$$

Lemma 3.8 *Let $P \in \mathcal{P}_{n,m}$ such a property, that every $G \in \min(P)$ has exactly $k \leq \frac{1}{2}n$ vertices of positive degree in colour class V. Then*

$$C^R(P) \geq \frac{1}{6}m \cdot n.$$

Proof. We consider the reduced graph property $P' = red_V(P|n+1-k)$. Obviously, P' is non-trivial and $\min(P')$ contains up to isomorphy exactly one graph. This graph has exactly one vertex with positive degree (d) in the colour class V'. We distinguish two cases.

Case 1. Assume that $d \leq \frac{2}{3}m$. Since the minimal graphs of P' and P'^D can't be packed as bipartite graphs, any $G \in \min(P'^D)$ has at least $(n+1-k)\cdot(m+1-d) \geq \frac{1}{6}m \cdot n$ edges. Hence Theorem 2.1. implies the required lower bound.

Case 2. If $d > \frac{2}{3}m$ then let us consider the induced property $ind_W(P'|d)$ on colour classes of size $n+1-k$ and d, respectively. Since $ind_W(P'|d) = S_{n+1-k,d}$, Lemma 3.7. yields the statement.

Lemma 3.9 *The randomized decision tree complexity of any subgraph isomorphism property $P_G \in \mathcal{P}_n$ with width at most $\frac{2}{3}n$ fulfilles*

$$C^R(P_G) \geq \frac{1}{24}(n^2 - 1).$$

Proof. Depending on $\text{width}(P_G)$ and $\text{trace}(P_G)$ we distinguish six cases.

Case 1. If $\frac{1}{2}n \leq width(P_G) \leq \frac{2}{3}n$ and $\text{trace}(P_G) \geq \frac{1}{12}n$, then assertion (i) of Lemma 3.3. implies the lower bound.

Case 2. If $\frac{1}{2}n \leq \text{width}(P_G) \leq \frac{2}{3}n$ and $\text{trace}(P_G) < \frac{1}{12}n$, then assertion (ii) of Lemma 3.3. implies the lower bound.

Case 3. If $K^*_{\lfloor n/2 \rfloor} \in P_G$, then $\text{width}(P_G) + \text{trace}(P_G) \leq \frac{1}{n}$. Therefore, by assertion (ii) of Lemma 3.3., we obtain $C^R(P_G) \geq \frac{1}{8}n^2 \geq \frac{1}{24}n^2$.

So far we have considered all the cases, when P_G can't be reduced to a non-trivial bipartite graph property $\text{bipart}(P_G)|\lfloor\frac{n}{2}\rfloor, \lceil\frac{n}{2}\rceil)$.

Case 4. If $\frac{n}{2} \leq \text{width}(P_G) < \frac{n}{2}$, then we can apply assertion (iii) of Lemma 3.3. for $\varepsilon = \frac{1}{2}$ and get the required lower bound.

Case 5. If $\text{width}(P_G) < \frac{n}{4}$ and $\text{trace}(P_G) \geq \frac{5}{6}n$, then G contains $\text{width}(P_G)$ vertices with degree at least $\frac{5}{9}n$. In our reduction to the bipartite graph property $\text{bipart}P_G|\lfloor\frac{n}{2}\rfloor, \lceil\frac{n}{2}\rceil)$ we have to put them all into V. On the other hand, these vertices build a covering of the graph G. Hence G contains no edge independent of this vertex set. Therefore any minimal graph of the property $\text{bipart}(P_G|\lfloor\frac{n}{2}\rfloor, \lceil\frac{n}{2}\rceil)$ has exactly $\text{width}(P_G)$ vertices of positive degree in V and Lemma 3.8. implies

$$C^R(P_G) \geq (bipart(P_G|\lfloor\frac{n}{2}\rfloor, \lceil\frac{n}{2}\rceil)) \geq \frac{1}{24}\cdot(n^2-1)$$

Case 6. If $\text{width}(P_G) < \frac{n}{4}$ and $\text{trace}(P_G) < \frac{5}{9}n$, then by assertion (ii) of Lemma 3.3., we get that

$$C^R(P_G) \geq \frac{1}{2}\cdot\frac{3}{4}n\cdot(\frac{3}{4}n - \frac{5}{9}n) \geq \frac{1}{24}n^2,$$

which completes the proof.

The following statement is an immediate consequence of this theorem and generalizes the results of Yao [6].

Assertion 3.10 *For every $\varepsilon > 0$ we can find a $\lambda > 0$ which depends only on ε, such that the randomized decision tree complexity of any subgraph isomorphism property $P_G \in \mathcal{P}_n$ with $\overline{d}(G) \leq \varepsilon$ fulfilles:*

$$C^R(P_G) \geq \lambda(\varepsilon) \cdot n^2.$$

After finishing this manuscript the author has learnt that M. Karpinski et al [2] independently proved Theorem 3.5.

References

[1] P. Hajnal, Complexity of graph properties, An $\Omega n^{\frac{4}{3}}$) lower bound on the randomized complexity of graph properties. Combinatorica, 11(2) (1991), 131-143.

[2] M. Karpinski, V. King, M. Szegedy, personal communication, 1990.

[3] V. King, Lower bounds on the complexity of graph properties, Proc. 20th ACM STOC (1988), 468-476.

[4] R. Rivest and Vuillemin, On recognizing graph properties from adjacency matrices, Theor. Comp. Sci. 3 (1976), 371-384.

[5] M. Saks and A. Widgerson, Probabilistic Boolean decision trees and the complexity of evaluating game trees, Proc. 27th IEEE FOCS (1986), 29-38.

[6] A. Yao, Probabilistic computation: towards a unified measure of complexity, Proc. 18th IEEE FOCS (1977), 222-227.

[7] A. Yao, Lower bounds to randomized algorithms for graph properties, Proc. 28th IEEE FOCS (1987), 393-400.

Received April 25, 1989.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

On the interaction between closure operations and choice functions with applications to relational databases*

János Demetrovics † Gusztáv Hencsey† Leonid Libkin‡
Ilya Muchnik§

Abstract

A correspondence between closure operations and special choice functions on a finite set is established. This correspondence is applied to study functional dependencies in relational databases.

1 Introduction

Having been introduced in connection with some topological problems, *closure operations* were applied in various branches of mathematics. In the last years they were successfully applied to study so-called *functional dependencies* (FDs for short) in relational databases. Now we recall some definitions and facts; they can be found in [DK],[DLM1].

Let $U = \{a_1, \ldots, a_n\}$ be a finite set of *attributes* (e.g. name, age etc.) and $W(a_i)$ the domain of a_i. Then a subset $R \subseteq W(a_i) \times \ldots \times W(a_n)$ is called a *relation over* U.

A *functional dependency* (FD) is an expression of form $X \longrightarrow Y$, $X, Y \subseteq U$. We say that FD $X \longrightarrow Y$ holds for a relation R if for every two elements of R with identical projections onto X, the projections of t hese elements onto Y also coincide. According to [Ar], the family $\mathcal{F}$ of all FD's that hold for R satisfies the properties (F1)-F4):

(F1) $(X \longrightarrow X) \in \mathcal{F}$;
(F2) $(X \longrightarrow Y) \in \mathcal{F}$ and $(Y \longrightarrow V) \in \mathcal{F}$ imply $(X \longrightarrow V) \in \mathcal{F}$;

*Research partially supported by HungarianNational Foundation for Scientific Research, Grant 2575.

†Computer and Automation Institute, P.O.Box 63, Budapest H-1518, Hungary; E-mail: h935dem@ella.hu and h103hen@ella.hu.

‡Department of Computer and Information Science, University of Pennsylvania, Philadelphia PA 19104, USA; E-mail: libkin@saul.cis.upenn.edu.

§24 Chestnut St., Waltham MA 02145, USA; E-mail: ilya@darwin.bu.edu.

(F3) $(X \longrightarrow Y) \in \mathcal{F}$ and $X \subseteq V, W \subseteq Y$ imply $(V \longrightarrow W) \in \mathcal{F}$;
(F4) $(X \longrightarrow Y) \in \mathcal{F}$ and $(V \longrightarrow W) \in \mathcal{F}$ imply $(X \cup V \longrightarrow Y \cup W) \in \mathcal{F}$.

Conversely, given a family of FD's satisfying (F1)-(F4) (so-called *full family*), there is a relation R over U generating exactly this family of FD's, see [Ar] and also [BDFS] for a constructive proof.

We shall write a_i instead of $\{a_i\}$ throughout the paper. Let R be a relation over $U, X \subseteq U$ and put $L_R(X) = \{a \in U | X \longrightarrow a$ holds for $R\}$. Then L_R satisfies

(C1) $X \subseteq L_R(X)$;
(C2) $X \subseteq Y \Longrightarrow L_R(X) \subseteq L_R(Y)$;
(C3) $L_R(L_R(X)) = L_R(X)$,

i.e. L_R is a *closure operation*. Note that the properties (C1)-(C3) may be concisely expressed as $X \subseteq L_R(Y)$ iff $L_R(X) \subseteq L_R(Y)$. Given a closure L (sometimes we shall omit the word "operation"), there is a relation R over U with $L = L_R$, see [De1].

A set $X \subseteq U$ is called *closed* (w.r.t. a closure L) if $L(X) = X$. Let $Z(L)$ stand for the family of all closed sets w.r.t. L. Then

(S1) $U \in Z(L)$,
(S2) $X, Y \in Z(L)$ implies $X \cap Y \in Z(L)$,

i.e. $Z(L)$ is a *semilattice*. Given a semilattice $Z \subseteq 2^U$ define $L(X) = \cap\{Y | X \subseteq Y, Y \in Z\}$. Then L is a closure with $Z(L) = Z$. Therefore, we can think of semilattices providing an equivalent description of closures and full families of FD's.

A closure is an extensive operation ($X \subseteq L(X)$). The operations satisfying the reverse inclusion (called *choice functions*) were also widely studied in connection with the theory of rational behaviour of individuals and groups, see [AM],[Ai],[Mo]. We give some necessary definitions.

A mapping $C : 2^U \longrightarrow 2^U$ satisfying $C(X) \subseteq X$ for every $X \subseteq U$, is called a choice function. U is interpreted as a set of alternatives, X as a set of alternatives given to the decision-maker to choose the best and $C(X)$ a s a choice of the best alternatives among X.

There were introduced some conditions (or properties) to characterize the rational behaviour of a decision-maker. The most important conditions are the following (see [AM],[Ai],[Mo]):
Heredity ($\underline{H}$ for short):

$$\forall X, Y \subseteq U : X \subseteq Y \Longrightarrow C(Y) \cap X \subseteq C(X);$$

Concordance ($\underline{C}$ for short):

$$\forall X, Y \subseteq U : C(X) \cap C(Y) \subseteq C(X \cup Y);$$

Out casting ($\underline{O}$ for short):

$$\forall X, Y \subseteq U : C(X) \subseteq Y \subseteq X \Longrightarrow C(X) = C(Y);$$

Monotonicity ($\underline{M}$ for short):

$$\forall X, Y \subseteq U : X \subseteq Y \Longrightarrow C(X) \subseteq C(Y).$$

Let P be a *binary relation* on U, i.e. $P \subseteq U \times U$. Let $C_P(X) = \{a \in X | (\nexists b \in X : (b, a) \in P)\}$.

One of the central results of the theory of choice functions states that a choice function can be represented as C_P for some P iff it satisfies $\underline{H}$ and $\underline{C}$.

Given a closure operation L, we can define choice functions $C(X) = L(U - X) \cap X$ and $C(X) = U - L(U - X)$. In Section 2 we characterize the choice functions of the second type as satisfying $\underline{M}$ and $\underline{O}$. In the other sections we use this correspondence to transfer the properties of choice functions to closures and to apply them to the study of FD's. In Section 3 we use the logical representation of choice functions (see [VR],[Li1]) to construct a similar representation and characterization of closure operations.

In Section 4 new properties of closure operations are obtained and studied by new properties being added to $\underline{M}$ and $\underline{O}$.

Finally, in the Section 5, we use choice functions to construct a structural representation for so-called *functional independencies* (cf. [Ja]) in the same way as closures were used to represent FD's.

2 The main correspondence

Let L be a closure operation. Define two choice functions associated with L as follows:

$$C_L(X) = L(U - X) \cap X,$$

$$C^L(X) = U - L(U - X), X \subseteq U.$$

Note that both C_L and C^L uniquely determine the closure L, in fact, $L(X) = X \cup C_L(U - X)$ and $L(X) = U - C^L(U - X)$. For every $X \subseteq U$ the sets $C_L(X)$ and $C^L(X)$ form a partition of X, i.e. $C_L(X) \cap C^L(X) = \emptyset$ and $C_L(X) \cup C^L(X) = X$.

Theorem 1 *The mapping $L \longrightarrow C^L$ establishes a one-to-one correspondence between the closure operations and the choice functions satisfying $\underline{O}$ and $\underline{M}$.*

Proof. Let L be a closure operation. We prove that C^L satisfies $\underline{M}$ and $\underline{O}$.

Let $x \in C^L(X)$ and $X \subseteq Y$. Then $x \notin L(U - X)$ and since $U - Y \subseteq U - X$, we have $x \notin L(U - Y)$, i.e. $x \in C^L(Y)$. Hence, C^L satisfies $\underline{M}$.

Let $X \subseteq U$. Then $L(L(U-X)) = L(U-X)$. Using $L(U-X) = U - C^L(X)$, we obtain that $U - C^L(U-(U-C^L(X)) = U - C^L(X)$, i.e. $C^L(C^L(X)) = C^L(X)$. Now let $C^L(X) \subseteq Y \subseteq X$; Since C^L satisfies $M, C^L(C^L(X)) \subseteq C^L(Y)) \subseteq C^L(X)$ and $C^L(X) = C^L(Y)$. Therefore, C^L satisfies $\underline{O}$.

Let C be a choice function satisfying $\underline{O}$ and $\underline{M}$. Consider $L(X) = U - C(U-X)$. We prove that L is a closure. Clearly, $X \subseteq L(X)$. If $X \subseteq Y$ and $x \in L(X)$, then $x \notin C(U-X)$ and $x \notin C(U-Y)$, i.e. $x \in L(Y)$. Since C satisfies $O, C(C(U-X)) = C(U-X)$. Applying $C(U-X) = U - L(X)$ we obtain $L(L(X)) = L(X)$. Hence, L is a closure and $C^L = C$.

To finish the proof, note that the mapping $L \longrightarrow C^L$ is injective, because for two distinct closures L_1 and L_2 with $L_1(X) \neq L_2(X)$ one has $C^{L_1}(U-X) \neq C^{L_2}(U-X)$. The theorem is proved.

□

Let K be a property of choice functions. We say that a choice function C satisfies $\overline{K}$ if its complement $\overline{C}$ satisfies K. (The complementary function $\overline{C}$ of C is defined as follows: $\overline{C}(X) = X - C(X)$ for $X \subseteq U$.)

Corollary 1 *The mapping $L \longrightarrow C_L$ establishes one-to-one correspondence between the closure operations and the choice functions satisfying $\underline{H}$ and $\underline{O}$.*

Proof. It follows from the facts that C_L and C^L are complementary choice functions and that $\underline{H} = \overline{M}, \underline{M} = \overline{H}$, see [Ai].

□

3 On logical representation of closure operations and choice functions

The family of all choice functions on U equipped with the operations $\cup, \cap$ and $^-$, is a Boolean algebra. Logical representation of the choice functions was introduced in [VR] to show that this Boolean algebra is isomorphic to one consisti ng of tuples of n Boolean functions, each depending on at most $n-1$ variables.

Let $U = \{a_1, \ldots, a_n\}$, $X \subseteq U$. Define

$$\beta_i(X) = \begin{cases} 1, & a_i \in X, \\ 0, & a_i \notin X, \end{cases}$$

$$\begin{aligned} \beta^i(X) &= (\beta_1(X), \ldots, \beta_{i-1}(X), \beta_{i+1}(X), \ldots, \beta_n(X)) \text{ and} \\ \beta^Z(X) &= (\beta_{i_1}(X), \ldots, \beta_{i_k}(X)) \end{aligned}$$

where $\{a_{i_1}, \ldots, a_{i_k}\} = U - Z$ and $i_1 < \ldots < i_k$.

Definition [VR]. *A family $\langle f_1^C, \ldots f_n^C \rangle$ of Boolean functions, each depending on $n-1$ variables, is called a first logical form of a choice function C if for every $a_i \in U$ and $X \subseteq U$:*

$$a_i \in C(X) \text{ iff } a_i \in X \text{ and } f_i^C(\beta^i(X)) = 1.$$

Definition [Li1]. *A family $\langle f_\emptyset^C, \ldots, f_U^C \rangle$ of Boolean functions indexed by subsets of U, is called a second logical form of a choice function C if for every $Z, X \subseteq U$:*

$$Z = C(X) \text{ iff } Z \subseteq X \text{ and } f_Z^C(\beta^Z(X)) = 1.$$

Note that f_Z^C depends on $n - |Z|$ variables.

Each logical form uniquely determines a choice function. By [VR], every tuple of Boolean functions, each depending on $n-1$ variables, is a first logical form of some choice function, moreover, $C \longrightarrow \langle f_1^C, \ldots, f_n^C \rangle$ is an isomorphism of Boolean algebras.

A family $\langle f_\emptyset, \ldots, f_U \rangle$, f_Z depends on $n - |Z|$ variables, is a second logical form of some choice function iff for each $Z \subseteq U$ the set $\{f_Z(\beta^Z(X)) : Z \subseteq X\}$ contains a unique one.

Let L be an operation satisfying (C1), i.e. $X \subseteq L(X)$ for all $X \subseteq U$. We can introduce two logical forms as before.

Definition. *A family $\langle f_1^L, \ldots, f_n^L \rangle$ of Boolean functions, each depending on at most $n-1$ variables, is called a first logical form of L if for every $a_i \in U$ and $X \subseteq U$:*

$$a_i \in L(X) \text{ iff } a_i \in X \text{ or } f_i^L(\beta^i(X)) = 1.$$

Let $Z = \{a_{i_1}, \ldots, a_{i_k}\}, i_1 < \ldots < i_k$, and $\beta_Z(X) = (\beta_{i_1}(X), \ldots, \beta_{i_k}(X))$.

Definition. *A family $\langle f_\emptyset^L, \ldots f_U^L \rangle$ of Boolean functions indexed by subsets of U, f_Z^L depends on $|Z|$ variables, is called second logical form of L if for every $Z, X \subseteq U$:*

$$Z = L(X) \text{ iff } X \subseteq Z \text{ and } f_Z^L(\beta_Z(X)) = 1.$$

We use these logical forms to characterize the closure operations among all the operations satisfying (C1).

Theorem 2 *Let L satisfy (C1). Then L is a closure operation iff all the functions $f_i^L, i = 1, \ldots, n$; $f_Z^L, Z \subseteq U$, are monotonic.*

Proof. Since $a_i \in L(X)$ iff $a_i \in X$ or $a_i \in C_L(U - X)$, we have $f_i^L(\beta^i(X)) = f_i^{C_L}(\beta^i(U - X))$, i.e. $\overline{f_i^L} = (f_i^{C_L})^*$, where f^* stands for the dual function. Analogously, we obtain that $\overline{f_Z^L} = (f_{U-Z}^{C^L})^*$ (note that f_Z^L and $f_{U-Z}^{C^L}$ depend on the same variables). Since $f_i^{C_L} = \overline{f_i^{C^L}}$ theorem 1 and the following facts imply the theorem: (1) C^L satisfies $\underline{M}$ iff all the functions $f_i^{C^L}$, $i = 1, \ldots, n$, a re monotonic (cf. [VR]);

(2) C^L satisfies $\underline{O}$ iff all the functions $\overline{f}_Z^{C^L}$, $Z \subseteq U$, are monotonic (cf. [Li1]). The theorem is proved.

□

Remark. A set of attributes $X \subseteq U$ is called a *candidate key* (w.r.t. a relation R) if $L_R(X) = U$ and for every $Y \subset X : L_R(Y) \neq U$. The problem of finding the candidate keys (or a candidate key) is one of the most important problems in the theory of relational databases, see e.g. [BDFS],[De2]. According to the previous theorem, the candidate keys are exactly the lower units of monotonic function $f_U^{L_R}$. Hence, we can apply a recognition algorithm for monotonic Boolean functions to construct an algorithm of finding the candidate keys. (Note that if we are given a set of FD's, we can calculate a value $f_U^{L_R}$ in polynomial time in the size of the set of FDs. However, the problem of finding all the candidate keys is NP-hard, see [BDFS]).

Some other aspects of the applications of recognition of monotonic Boolean functions to the study of choice functions satisfying $\underline{M}$ and $\underline{O}$ (and, hence, closure operations) can be found in [Li2].

4 On the properties of closures induced by the properties of choice functions

In this section we consider the closures for which choice functions C_L and C^L defined in Section 2 satisfy some additional properties. Note that in the theory of choice functions such properties are ussually studied in some fixed combinations. These combinations explain the use of C_L and C^L. E.g., the property $\underline{C}$ (Concordance) is usually studied together with $\underline{H}$ (see [Ai], [AM],[Mo],[Li1]). Thus, studying this property we consider C_L (moreover, the property $\underline{C}$ implies monotonicity and there is no reason to consider C^L).

Property $\underline{C}$. As it was mentioned, we consider the functions C_L.

Let L be a closure and $\mathcal{F}_L$ a corresponding full family of FD's. Recall that an FD $X \longrightarrow Z$ is called *nontrivial* [De2],[DLM1] if $X \cap Z = \emptyset$. Let P_6 stand for the (Post) class consisting of conjunctions and constants, cf. [Po].

Proposition 1 *Let L be a closure operation on U. The following are equivalent:*

1) C_L satisfies the property $\underline{C}$;

2) $L(X) \cap L(Y) - (X \cup Y) \subseteq L(X \cap Y)$ for all $X, Y \subseteq U$;

3) If $X \longrightarrow Z$ and $Y \longrightarrow Z$ are nontrivial FD's from $\mathcal{F}_L$, then $X \cap Y \longrightarrow Z \in \mathcal{F}_L$;

4) $(X \longrightarrow a) \in \mathcal{F}_L$ iff $U - \{a, b\} \longrightarrow a$ for all $b \notin X$, where $a \notin X$;

5) For all $i = 1, \ldots, n : f_i^L \in P_6$.

Proof. 1 $\longrightarrow$ 2. Let C_L satisfy $\underline{C}$. Then for all $X, Y \subseteq U : C_L(U - X) \cap C_L(U - Y) \subseteq C_L(U - X \cap Y)$. Using $C_L(Z) = L(U - Z) \cap Z$ we obtain $L(X) \cap L(Y) - (X \cup Y) \subseteq L(X \cap Y) - (X \cap Y)$. Hence, 2 hold s.

$2 \longrightarrow 3$. Let $X \longrightarrow Z$ and $Y \longrightarrow Z$ be nontrivial FD's from $\mathcal{F}_L$. Then so are $X \longrightarrow a$ and $Y \longrightarrow a$ for all $a \in Z$. Since $a \in L(X) \cap L(Y) - (X \cup Y)$, we have that $a \in L(X \cap Y)$, i.e. $X \cap Y \longrightarrow a \in \mathcal{F}_L$. Then by (F4) $X \cap Y \longrightarrow Z \in \mathcal{F}_L$.

$3 \longrightarrow 1$. Let 3) hold and $a \in C_L(X) \cap C_L(Y)$, $X, Y \subseteq U$. Then $U - X \longrightarrow a \in \mathcal{F}_L$ and $U - Y \longrightarrow a \in \mathcal{F}_L$ and both FD's are nontrivial. Hence, $U - (X \cup Y) \longrightarrow a \in \mathcal{F}_L$ and $a \in L(U - (X \cup Y))$. Since $a \in (X \cup Y)$, we have $a \in C_L(X \cup Y)$. Therefore, C_L satisfies $\underline{C}$.

$1 \longleftrightarrow 4$. Let $a \notin X$. Then $X \longrightarrow a \in \mathcal{F}_L$ iff $a \in C_L(U - X)$, and $U - \{a, b\} \longrightarrow a \in \mathcal{F}_L$ iff $a \in C_L(\{a, b\})$. Hence, 4) is equivalent to: $a \in C_L(Z)$ iff $a \in C_L(\{a, b\})$ for all $b \in Z$. According to [AM],[Mo] the last property holds iff C_L satisfies $\underline{C}$.

$1 \longleftrightarrow 5$. Since C_L satisfies $\underline{H}$, it satisfies $\underline{C}$ iff all the functions $f_i^{C_L} i = 1, \ldots, n$ can be represented as $\overline{f}^*$, where $f \in P_6$, see [VR],[Li1]. Since $f_i^L = \overrightarrow{f_i^{C_L}}$, we have that C_L satisfies $\underline{C}$ iff $f_i^L \in P_6$ for all i. The proposition is proved.

□

Property of submission. This property was introduced in [Li1] as a dual form of $\underline{C}$. We say that a choice function satisfies the submission property ($\underline{S}$ for short) if

$$\forall X, Y \subseteq U : C(X \cap Y) \subseteq C(X) \cup C(Y).$$

Recall that a closure is called *topological* if $L(X \cup Y) = L(X) \cup L(Y)$ for all $X, Y \subseteq U$.

Let S_6 stand for the class of Boolean functions consisting of disjunctions and constants, cf. [Po].

Proposition 2 *Let L be a closure operation. Then the following are equivalent:*

1) C_L satisfies $\underline{S}$;

2) L is a topological closure;

3) $X \longrightarrow a \in \mathcal{F}_L$ iff $b \longrightarrow a \in \mathcal{F}_L$ for some $b \in X$;

4) For all $i = 1, \ldots, n : f_i^L \in S_6$.

Proof. $1 \longrightarrow 2$. Let C_L satisfy $\underline{S}$. Then for all $X, Y \subseteq U : L(X \cup Y) = X \cup Y \cup C_L(U - X \cup Y) = X \cup Y \cup C_L((U - X) \cap (U - Y)) \subseteq (X \cup C_L(U - X)) \cup (Y \cup C_L(U - Y)) = L(X) \cup L(Y)$. Since (C2) holds, $L(X) \cup L(Y) \subseteq L(X \cup Y)$, i.e. L is topological.

$2 \longrightarrow 1$. Let L be topological. Then for all $X, Y \subseteq U : C_L(X \cap Y) = L(U - X \cap Y) \cap X \cap Y = L((U - X) \cup (U - Y)) \cap X \cap Y \subseteq (L(U - X) \cup L(U - Y)) \cap X \cap Y \subseteq (L(U - X) \cap X) \cup (L(U - Y) \cap Y)) = C_L(X) \cup C_L(Y)$, i.e. C_L satisfies $\underline{S}$.

$2 \longleftrightarrow 3$. It was proved in [DLM2].

$1 \longleftrightarrow 4$. According to [Li1], C_L satisfies $\underline{S}$ iff for all $i = 1, \ldots, n : (f_i^{C_L})^* \in S_6$, i.e. iff $f_i^L \in S_6$. The proposition is proved.

□

The topological closures are known to have simple matrix representations. Consider two binary relations P_L and T_L on U as follows:

$(a_i, a_j) \in P_L$ iff every closed subset X (w.r.t. L) either contains a_j or does not contain a_i.

$(a_i, a_j) \in T_L$ iff $a_j \in L(a_i)$.

For a closure L, P_L is a reflexive relation. Given a reflexive relation P suppose that $L(X)$ is the intersection of all $Y \supseteq X$ such that for all $(a_i, a_j) \in P$ either $a_i \notin Y$ or $a_j \in Y$. Then L thus constructed is a topological closure with $P_L = P$, see [DLM2].

For a topological closure L, T_L is a transitive binary relation. Conversely, given a transitive binary relation T, define $L(X) = X \cup \{a \in U | \exists b \in X : (b, a) \in T\}$. Then L is a topological closure with $T_L = T$. Moreover, T_L is t he minimal transitive binary relation containing P_L, see [DLM2]

It is known that the choice functions satisfying $\underline{H}$ and $\underline{S}$ can be represented by binary relation as follows [Li1]:

$$C^P(X) = \{a \in X | \exists b \in X : (b, a) \in P \Longrightarrow \exists c \notin X : (c, a) \in P\}.$$

Hence, P thus constructed can be considered as a representation of a topological closure with $C_L = C^P$.

Proposition 3 *$C_L = C^{T_L}$ holds for any topological closure L.*

Proof. Let $a \in X$. Since T_L is reflexive, $a \in C^{T_L}(X)$ iff for some $c \notin X$: $(c, a) \in T_L$, i.e. iff $a \in L(c)$. Since L is topological, the last is equivalent to $a \in L(U - X) \cap X$, i.e. $a \in C_L(X)$.

□

Property of multi-valued concordance. This property also has been introduced in [Li1] in order to be studied together with the property $\underline{O}$.

A subset of $U \times 2^U$ was called in [AM] a *hyper-relation.* We will call a hyper-relation *correct* [Li1] if for every $X \subseteq U$ there is a unique $Y \subseteq X$ such that for all $a \in X - Y$ the pairs (a, Y) belong to the hype r-relation.

Proposition 4 *Let L be a closure operation. Then the following are equivalent:*

1. *C^L satisfies the property of multivalued concordance, i.e. if $Z = C^L(X) = C^L(Y)$ then $Z = C^L(X \cup Y)$;*
2. *For all $X, Y \subseteq U$: $L(X) = L(Y)$ implies $L(X) = L(X \cap Y)$;*
3. *For all $Z \subseteq U$: $f_Z^L \in P_6$;*
4. *For all $X \subseteq U$: $C^L(X) = Y$, where $(a, Y) \in D$ for all $a \in X - Y$ and D is a correct hyper-relation.*

Proof. The equivalence of 1 and 2 is evident. The equivalences 1 $\longleftrightarrow$ 3 and 1 $\longleftrightarrow$ 4 follow from [Li1].

□

5 Structural representation of functional independencies

Let R be a relation over U. We say that a *functional independency* (FID for short) $X \longrightarrow Y$ holds for R if there are two elements of R with coinciding projections onto X and distinct projections onto Y (i.e. FD $X \longrightarrow Y$ does not hold), see [Ja]. A review of properties of FID's can be found in [Ja]. In this section we construct the representations of FID's via operations on a power set and semilattices.

Let R be a relation and $\mathcal{F}I_R$ the family of all FID's that hold for R. A family $\mathcal{F}I$ of FID's is called *full* if for some relation R one has $\mathcal{F}I = \mathcal{F}I_R$.

Given a full family $\mathcal{F}I$, define for $X \subseteq U$ $C_{\mathcal{F}I}(X) = \{a \in X | (U - X) \longrightarrow a \in \mathcal{F}I\}$. Conversely, given a choice function C, define a family of FID's $\mathcal{F}I_C$ as follows:

$$X \longrightarrow Y \in \mathcal{F}I_C \text{ iff } Y \subseteq C(U - X).$$

Let C be a choice function. Define $\mathcal{L}(C) = \{X \subseteq U | C(X) = X\}$. For a join-semilattice $\mathcal{L}$, $(\mathcal{L} \subseteq 2^U, \emptyset \in \mathcal{L}, X, Y \in L \Longrightarrow X \cup Y \in L)$ define $C_{\mathcal{L}}$ as follows:

$$C_{\mathcal{L}}(X) = \cup(Y | Y \subseteq X, Y \in \mathcal{L}).$$

Theorem 3 *a) The mappings $\mathcal{F}I \longrightarrow C_{\mathcal{F}I}$ and $C \longrightarrow \mathcal{F}I_C$ establish mutually inverse one-to-one correspondences between full families of FID's and choice functions satisfying $\underline{M}$ and $\underline{O}$.*

b) The mappings $C \longrightarrow \mathcal{L}(C)$ and $\mathcal{L} \longrightarrow C_{\mathcal{L}}$ establish mutually inverse one-to-one correspondences between choice functions satisfying $\underline{M}$ and $\underline{O}$ and join-semilattices.

Proof. a) Let $\mathcal{F}I = \mathcal{F}I_R$ be a full family of FID's. Then $a \in C_{\mathcal{F}I}(X)$ iff $a \notin L_R(U - X)$, i.e. $C_{\mathcal{F}I}(X) = U - L_R(U - X)$ and C satisfies $\underline{O}$ and $\underline{M}$ by theorem 1.

Let C satisfy O and M. Then $C = C^L$ for some closure L, and $X \longrightarrow Y \in \mathcal{F}I_C$ iff $Y \cap L(X) = \emptyset$, i.e. $(X \longrightarrow Y) \notin \mathcal{F}_L$. Hence $\mathcal{F}I_C$ is a full family. Moreover, $a \in C_{\mathcal{F}I_C}(X)$ iff $(U - X) \longrightarrow a \in \mathcal{F}I_C$ iff $a \in C(X)$. Part a is proved.

b) Let L be a closure. Then $\mathcal{L}(C^L) = \{X \subseteq U | C^L(X) = X\} = \{X \subseteq U | L(U - X) = U - X\} = \{X \subseteq U | U - X \in Z(L)\}$. Hence, part b follows from theorem 1 and the well-known correspondence bet ween (meet)-semilattices and closure operations, see [DK],[DLM1]. The theorem is proved.

□

The last question to be considered is as follows: when is a full family of FID's also a full family of FD's? In other words, when is a closure operation $L(X) = X \cup \{a \notin X | X \longrightarrow a \in \mathcal{F}I_R\}$?

Proposition 5 *Let R be a relation over U. Then the following are equivalent:*

1. *$L(X) = X \cup \{a \notin X | X \longrightarrow a \in \mathcal{F}I_R\}$ is a closure operation;*

2. There is $Z \subseteq U$ such that $L_R(X) = X \cup Z$ for all $X \subseteq U$.

Proof. Let $L(X) = X \cup \{a \notin X | X \longrightarrow a \in \mathcal{F}I_R\}$ be a closure. Then C^L satisfies $\underline{H}$ (see theorem 1) and since C^L satisfies $\underline{M}$ we have that for some $V \subseteq U : C^L(X) = X \cap V$ for all $X \subseteq U$, see [AM]. Therefore, for $Z = U - V$ one has $L_R(X) = X \cup Z$.

Conversely, if L_R is as in 2, then $L(X) = X \cup \{a \notin X | X \longrightarrow a \in \mathcal{F}I_R\} = X \cup \{a \notin X | X \longrightarrow a \notin \mathcal{F}_R\}$ is obviously a closure operation. The proposition is proved.

□

References

[Ai] M.A. Aizermann, New problems in the general choice theory (Review of a research trend), J. Social Choice and Welfare 2 (1985), 235-282.

[AM] M.A. Aizerman and A.V. Malishevski, General theory of best variants choice: Some aspects, IEEE Trans. Automat. Control 26 (1981), 1030-1041.

[Ar] W.W. Armstrong, Dependency structure of data-base relationship, Information Processing 74, North Holland, Amsterdam, (1974), 580-583.

[BDFS] C. Beeri, M. Dowd, R. Fagin and R. Statman, On the structure of Armstrong relations for functional dependencies, J. ACM 31 (1984), 30-46.

[De1] J. Demetrovics, Candidate keys and antichains, SIAM J. Alg. Disc. Meth. 1 (1980), 92.

[De2] J. Demetrovics, On the equivalence of candidate keys with Sperner systems, Acta Cybernetica 4 (1979), 247-252.

[DK] J. Demetrovics and G.O.H. Katona, Extremal combinatorial problems of database models, MFDBS 87, Springer LNCS 305 (1988), 99-127.

[DLM1] J. Demetrovics, L.O. Libkin and I.B. Muchnik, Functional dependencies and the semilattice of closed classes, MFDBS 89, Springer LNCS 364 (1989), 136-147.

[DLM2] J. Demetrovics, L.O. Libkin and I.B. Muchnik, Closure operations and database models (in Russian) to appear in Kibernetika, Kiew.

[Ja] J. M. Janas, Covers for functional independencies, MFDBS 89, Springer LNCS 364(1989), 254-268.

[Li1] L.O. Libkin, Algebraic methods for construction and analysis of the choice function classes (in Russian), Individual Choice and Fuzzy Sets, Trudy VNIISI, Moscow 14(1987), 46-53.

[Li2] L.O. Libkin, Recognition of choice functions (in Russian), Automatika i Telemehanika, 1988, No. 10, p. 128-132. English Translation in: J. Automation and Remote Control.

[Mo] H. Moulin, Choice functions over a finite set: A summary. IMA Preprint series, University of Minnesota, Minneapolis, 1984.

[Po] E. Post, Two-valued Iterative Systems, 1941.

[VR] T.M. Vinogradskaja and A.A. Rubchinski, Logical forms of choice functions (in Russian), Dokl. Akad. Nauk. SSR 254(1980), 1362-1366. English Translation in: Soviet Math. Dokl.

Received February 1, 1990.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

Normal Form Relation Schemes: A New Characterization*

János Demetrovics † Gusztáv Hencsey† Leonid Libkin‡
Ilya Muchnik§

Abstract

A new characterization of relational database schemes in normal forms is given. This characterization is based on the properties of the semilattice of closed sets of attributes. For the problems testing third and Boyce-Codd normal forms, which are known to be $\mathcal{NP}$-complete for relation schemes, this new characterization helps establish polynomial algorithms if the input is a relation (matrix) rather than a relation scheme. The problem of approximation of an arbitrary family of functional dependencies by one in a normal form is also addressed.

1 Introduction

The relational datamodel defined by E.F. Codd remains one of the most powerful database models. In this model a relation is just a matrix in which rows correspond to records and columns to attributes. Theoretical and practical aspects of this model have been studied over the past 20 years. Database design has always been and still is among the most important aspects attracting the attention of almost all database theorists. For relational databases, the design theory is based on the well-developed theory of dependencies and constraints. Functional dependencies, being the simplest and easiest to understand, underwent a deep investigation. Enormous number of papers on functional dependencies have been written, [10,11,18,22,23,24] being just examples of surveys referring to hundreds of other papers and books. Surprisingly enough, many issues in dependency theory, lying on the very surface, have not been paid attention to for many years. One of them is the lattice-theoretic approach to the study of functional dependecies. It was observed very early that families of functional dependencies correspond to closure operators and to semilattices, but very little has been done in order to bring the methods and tools of lattice

*Research partially supported by Hungarian Foundation for Scientific Research (OTKA), Grant 2575. While working on the revised version, L. Libkin was partially supported by NSF Grants IRI-86-10617 and CCR-90-57570 and ONR Grant NOOO14-88-K0634.

†Computer and Automation Institute, P.O.Box 63, Budapest H-1502, Hungary; E-mail: h935demOella.hu and h103henOella.hu.

‡Department of Computer and Information Science, University of Pennsylvania, Philadelphia PA 19104, USA; E-mail: libkinOsaul.cis.upenn.edu.

§24 Chestnut St., Waltham MA 02145, USA; E-mail: ilyaOdarwin.bu.edu.

theory to the database theory. The situation started changing a few years ago, and in a number of papers functional dependencies were investigated from the lattice theoretic point of view [4,6,7,25]. For example, an easily described relationship between relations and irreducible elements of the semilattice of closed sets made it possible to design a polynomial algorithm for a problem that is well-known to be $\mathcal{NP}$-complete if the input is a relation scheme rather than a relation, see [8,9].

Functional dependencies are closely related to *normalization* of relations or relations schemes. Being in a normal form, or normalized, means that a family of functional dependencies satisfies certain properties. The basic idea behind normalization is that a relational database must be unambigously reconstructed from some of its projections which are normalized. Databases in normal forms are easy to work with, and normal forms are well motivated from the practical point of view, see [5,22,24].

However, to the best of the authors' knowledge, no attempts have been made to apply lattice theoretic techniques, used for functional dependencies, to normalization. We think that doing so would benefit both normalization theory by looking at normalization from a new point of view, and lattice theoretic approach to the study of functional dependencies by extending it to normalization.

The main goal of this paper is to give a lattice theoretic characterization of relation schemes in normal forms, i.e. to describe normal form relation schemes by the semilattices of closed sets they generate. Doing only this would be of little interest. We prefer to view the characterization theorems as important tools in demonstrating advantages of the lattice theoretic approach. In this paper we are going to elaborate on two points:

- the lattice characterization of normal forms will enable us to prove that two problems related to normalization, which are known to be $\mathcal{NP}$-complete for relation schemes, are solved in polynomial time for relations (i.e. databases themselves);

- it will turn out that arbitrary families of functional dependencies can be approximated by normalized ones, and these approximations are effectively computable for relations; for relation schemes, however, it may take exponential time to find approximations.

Let us give a brief sketch of the rest of the paper. The next section contains all necessary definitions and facts. Most of them are standard, but some are not. We define all the concepts because we feel that a paper in an area where different people use slightly different terminology and completely different notation, must be self-contained.

Section 3,4 and 5 deal with the second, third and Boyce-Codd normal forms, respectively. (First normal form basically says that a database is just a relation. Therefore, functional dependency families can be characterized in one word – arbitrary). For each normal form we consider four problems:

- a lattice-theoretic characterization;

- closure properties in the lattice of families of functional dependencies;

- algorithms testing relations and relation schemes for this normal form and their complexity;

- approximation of arbitrary relation schemes by those in normal forms.

Characterisation theorems will be stated in the following form: a relation scheme is in normal form iff the semilattice of sets closed under the closure operation induced by the given scheme satisfies certain properties.

It is known that closure operations on an arbitrary set form a lattice [4,7]. We will show the properties of the subsets of this lattice corresponding to normal form schemes.

Before giving the results on complexity, recall that the *prime attribute problem* is to decide whether a given attribute is prime, i.e. belongs to a minimal key. It is not a complete description of the prime attribute problem for we did not indicate what the input is - a relation scheme or a relation. In the first case the problem is known to be $\mathcal{NP}$-complete [12,17]. However, it was shown in [8] that the problem becomes polynomial for relations. It was done by using the representation of irreducible elements of the semilattice of closed sets which can be obtained from a relation in polynomial time.

Two important problems related to normalization – 3NFTEST and BCNFTEST – are known to be $\mathcal{NP}$-complete [1,14]. They test whether a given relation scheme or its subscheme is in third or Boyce-Codd normal form. Using the techniques similar to those in [8] and our lattice characterization or normal forms, we shall prove that these problems can be solved in polynomial time if the input is a relation.

By *approximation* we mean finding a relation scheme that approximates a given one. "Approximates" should be explained here. First, the approximation must be taken from the class in which it is sought (otherwise the name would not be justified!). Second, it must be greater than the given scheme in some sense. Here we use the ordering on the families of functional dependencies (or closures) introduced and studied in [4,7] to define "greater". Finally, it is desired that approximation be unique. Uniqueness, as it will be shown, depends on closure properties of the given normal form, and can be guaranteed for third and Boyce-Codd normal forms.

If we want to find an approximation, we would like to know the complexity of an algorithm. It will turn out that the situation here resembles the one in the case of testing normal forms: for relations there exist polynomial algorithms, while for relation schemes the problems are superpolynomial[1], provided that $\mathcal{P} \neq \mathcal{NP}$.

2 Basic definitions and results

In this section, that we shall try to make as concise as possible, all definitions and facts to be used in the sequel will be given. Theorems in this section will have negative numbers so that our first result is theorem 1.

Let $\mathcal{U} = \{A_1, \ldots, A_n\}$ be a set of attributes. With each attribute A_i associate a domain of its values $dom(A_i)$. A *relation* over $\mathcal{U}$ is a subset of Cartesian product of all $dom(A_i)$'s. Relations will be usually denoted by R, possibly with indices. Alternatively, we can think of a relation R as being a set of maps $h : \mathcal{U} \to \bigcup_i dom(A_i), h(A_i) \in dom(A_i)$ rather than a set of tuples. This does not change the nature of relations, but often makes the notation easier. $R = \{h_1, \ldots, h_m\}$ means that R is a relation consisiting of m tuples/maps $h_1, \ldots, h_m$.

A *functional dependency* (FD for short) is an expression $X \to Y$, where $X, Y \subseteq \mathcal{U}$. If $A \in \mathcal{U}$, we shall write $X \to A$ instead of $X \to \{A\}$. A FD $X \to Y$ *holds* in a relation $R = \{h_1, \ldots, h_m\}$ if for any $h_i, h_j \in R$ the following holds: $\forall A \in Y : h_i(A) = h_j(A)$ whenever $\forall A \in X : h_i(A) = h_j(A)$. A family of FDs $F_R = \{X \to Y : X \to Y \text{ holds in } R\}$ is called a *full family of FDs*.

[1]That is, there are no polynomial algorithms that solve these problems.

Let $\mathcal{P}(U)$ be the powerset of U. We can think of $\rightarrow$ as being a binary relation on $\mathcal{P}(U)$, thus representing a family of functional dependencies as a binary relation (i.e. a set of pairs (X, Y)) as well. A subset F of $\mathcal{P}(U) \times \mathcal{P}(U)$ is called an *f-family* if the following (Armstrong's Axioms) hold:

(F1) $(X, X) \in F$;
(F2) $(X, Y) \in F, (Y, Z) \in F$ imply $(X, Z) \in F$;
(F3) $(X, Y) \in F, X \subseteq X', Y' \subseteq Y$ imply $(X', Y') \in F$;
(F4) $(X, Y) \in F, (Z, V) \in F$ imply $(X \cup Z, Y \cup V) \in F$.

For any binary relation $F \subseteq \mathcal{P}(U)$, F^+ stands for the minimal binary relation containing F and satisfying (F1)-(F4). The existence of F^+ is ensured by the fact that f-families are closed under intersection. F_R is an f-family for any relation R, i.e. $F_R^+ = F_R$.

A map $L : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ is called a *closure* if it is expanding, monotone and idempotent, i.e. $X \subseteq L(X), X \subseteq Y$ implies $L(X) \subseteq L(Y)$ and $L(L(X)) = L(X)$. For a binary relation F on $\mathcal{P}(U)$ define $L_F(X) = \{A \in U : (X\{A\}) \in F^+$. L_F thus defined is known to be a closure. If $F = F_R$, we write L_R instead of L_{F_R}.

A family of subsets $S \subset \mathcal{P}(U)$ is called a *(meet)-semilattice* if it is closed under intersection, i.e. $X, Y \in S$ implies $X \cap Y \in S$. Given a closure L, define $S_L = \{X \subseteq U : L(X) = X\}$ and $F_L = \{(X, Y) : Y \subseteq L(X)\}$. The elements of S_L are called *closed* sets. Given a semilattice S containing U, define a map L_S on $\mathcal{P}(U)$ by $L_S(X) = \bigcap\{Y : Y \in S, X \subseteq Y\}$.

Theorem -3 *a) $F \subseteq \mathcal{P}(U) \times \mathcal{P}(U)$ is an f-family iff there is a relation R such that $F_R = F$.*

b) The maps $F \rightarrow L_F$ and $L \rightarrow F_L$ defined above are mutually inverse and set up a 1-1 correspondence between closures and f-families on U.

c) The maps $L \rightarrow S_L$ and $S \rightarrow L_S$ defined above are mutually inverse and set up a 1-1 correspondence between closures on U and semilattices of subsets of U, containing $\{U\}$. □

This theorem shows that we do not have to redefine concepts, once introduced for families of FDs or relations or closures or semilattices, if we need their interpretations for other objects - they can be easily obtained from the 1-1 correspondences of theorem -3.

In the sequel, by *relation scheme* we shall mean a pair $\langle U, F \rangle$. All concepts defined for a relation scheme are automatically defined for any relation R by taking the relation scheme $\langle U, F_R \rangle$.

Given a relation scheme $\langle U, F \rangle$, a set $K \subseteq U$ is called a *key* if $K \rightarrow U \in F^+$ (equivalently, $L_F(K) = U$). A key is called *minimal* (sometimes *candidate*) if it contains no key as a proper subset. All minimal keys form an *antichain* (i.e. $K_1 \not\subseteq K_2$ for any two distinct minimal keys K_1, K_2) and vice versa: any antichain in $\mathcal{P}(U)$ can be represented as a family of minimal keys of a relation scheme or a relation over U.

Given a relation or a relation scheme, an attribute A is called *prime* if it belongs to a minimal key, and *nonprime* otherwise. The sets of prime and nonprime attributes will be denoted by U_p and U_n (or $U_p(F), U_p(L), U_p(R)$ etc. if R or F or L is not clear from the context).

Given a relation scheme $\langle U, F \rangle$, it is said to be in

- *Second Normal Form* (or 2NF for short) if for any minimal key K and a nonprime attribute A, $K' \rightarrow A \in F^+$ for no $K' \subset K$;

- *Third Normal Form* (or 3NF for short) if for any nonprime attribute A and X not containing A, X is a key whenever $X \rightarrow A \in F^+$;
- *Boyce-Codd Normal Form* (or BCNF for short) if X is a key whenever $X \rightarrow A \in F^+$ for $A \notin X$.

All the definitions given above are fairly standard. Now we introduce some terminology that appeared relatively recently in [4,7,8,9].

An *antikey* is a maximal non-key. In other words, let $K = \{K_1, \ldots, K_r\}$ be a family of minimal keys of a relation or a relation scheme. Then X is an antikey if $K_i \subseteq X$ for no i and X is maximal such. The set of antikeys will be denoted by K^{-1}.

Given a semilattice S, $M(S)$ stands for the set of (meet)-irreducible elements, i.e. such $X \in S$ that $X = Y \cap Z, Y, Z \in S$ implies either $Y = X$ or $Z = X$. Every element of a finite semilattice is an intersection of irreducibles. Maximal elements of $S - \{U\}$ are called *coatoms*. The set of coatoms is denoted by $CA(S)$.

Theorem -2 [8,9] $\bigcup K = U - \bigcap K^{-1}$. □

Theorem -1 [8,9] *Given a closure L, the set of antikeys it generates is $CA(S_L)$.* □

Given a relation $R = \{h_1, \ldots, h_m\}$, let $E_{ij} = \{A \in U : h_i(A) = h_j(A)\}$, and $E_R = \{E_{ij} : 1 \leq i < j \leq m\} \bigcup \{U\}$. E_R is called the *equality* set of R. It turns out that E_R contains all information about dependencies in R, i.e. L_R can be obtained from E_R by $L_R(X) = \bigcap \{Y : Y \in E_R, X \subseteq Y\}$. $CA(S_R)$ contains exactly the maximal sets from $E_R - \{U\}$ [8,9].

Let Cl_U be the set of all closures on U. Without loss of generality we shall also denote it by Cl_n if $|U| = n$. Define $\geq$ on Cl_n by letting $L_1 \geq L_2$ iff $L_1(X) \subseteq L_2(X)$ for all X (in other words, $L_1 \cdot L_2 = L_2$).

Theorem 0 [7] *Cl_n is a lattice in which infimum ($\wedge$) and supremum ($\vee$) are defined as follows: $L = L_1 \wedge L_2$ iff $S_L = S_{L_1} \bigcap S_{L_2}$, $L = L_1 \vee L_2$ iff $S_L = S_{L_1} \bigcup S_{L_2} \bigcup \{X \cap Y : X \in S_{L_1}, Y \in S_{L_2}\}$.* □

In fact, $\wedge$ and $\vee$ can be expressed directly, but for our purposes this semilattice definition suffices.

A subset of Cl_n closed w.r.t. $\wedge$ ($\vee$ or both $\wedge$ and $\vee$) is called a *meet-subsemilattice (join-subsemilattice* and *sublattice*) respectively.

Given $X \subseteq U$, let $Cl_n(X) = \{L \in Cl_n : U_p(L) = X\} \bigcup \{L^1\}$, where L^1 is the top element of Cl_n, i.e. $L^1(Y) = Y$ for any Y.

The last definition to be given in this section is that of *interval:* if $X \subseteq Y \subseteq U$, then $[X, Y]$ is the family of $Z \subseteq U$ such that $X \subseteq Z \subseteq Y$.

2.1 Second Normal Form

In this section we give a semilattice characterisation of the second normal form (2NF). The set $2NF_n \subseteq Cl_n$ of the closures generated by relation schemes in 2NF will be shown to be neither meet- nor join- subsemilattice of Cl_n. An approximation of a closure defined as the one generated by a 2NF relation scheme and having the same set of prime attributes will be shown to exist.

Let L be a closure. A closed set, which can be obtained as the closure of a proper subset of a minimal key, will be called *prime*. In other words, a closed set X is prime if $X = L(Y)$ where $Y \subset K$ and K is a minimal key.

Theorem 1 *Let R be a relation over $\mathcal{U}$. Then R is in 2NF iff $[X \cap \mathcal{U}_p, X] \subseteq S_R$ for any prime $X \subseteq \mathcal{U}$.*

Proof. Suppose $[X \cap \mathcal{U}_p, X] \subseteq S_R$ for all prime $X \subseteq \mathcal{U}$. Assume R is not in 2NF, i.e. for some $A \in \mathcal{U}_n$, a minimal key K and $K' \subset K$ one has $K' \to A \in F_R$ and $A \notin K'$. Let $X = L_R(K')$. Clearly, X is prime, $X \neq \mathcal{U}$ and $A \in X$. Since X is closed, $X - A \to A \in F_R$, and $X - A \notin S_R$ (if $X - A \in S_R, X - A$ is a closed set containing K' which is a subset of X). On the other hand, $A \notin \mathcal{U}_p$, and $X \cap \mathcal{U}_p \subseteq X - A \subseteq X$, i.e. $X - A \in [X \cap \mathcal{U}_p, X] \subseteq S_R$, a contradiction. Thus, R is in 2NF.

Conversely, let R be in 2NF. Take a prime X where $X = L(Y), Y \subset K, K$ a minimal key. Let $A \in \mathcal{U}_n \cap X$. Then $A \notin Y$. If $X - A \to A \in F_R$, then $Y \to A \in F_R$, which contradicts our assumption that R is in 2NF. Hence $X - A \to A \notin F_R$, and since X is closed, so is $X - A$. Since S_R is a semilattice, $[X - \mathcal{U}_n, X] = [X \cap \mathcal{U}_p, X] \subseteq S_R$. The theorem is proved.

Define $2NF_n \subseteq Cl_n$ to be the subset of Cl_n consisting of all closures induced by relation schemes in 2NF. This set does not have any particular structure as a subset of Cl_n, i.e. it is neither meet- nor join- subsemilattice, as the following examples show. Let $\mathcal{U} = \{A_1, \ldots, A_{11}\}$, and consider three semilattices: $S = \{\emptyset, A_1, A_2, A_3, \{A_1, A_2, A_4\}, \{A_1, A_3, A_5\}, \mathcal{U}\}$, $S_1 = S \bigcup \{\{A_1, A_2, A_4, A_6\}, \{A_1, A_3, A_5, A_7\}, A_{10}\}$, $S_2 = S \bigcup \{\{A_1, A_2, A_4, A_8\}, \{A_1, A_3, A_5, A_9\}, A_{11}\}$. Then both L_{S_1} and L_{S_2} are in $2NF_{11}$, but $L_S = L_{S_1} \wedge L_{S_2}$ is not. A counterexample in the case of $\vee$ is even simpler. Let $\mathcal{U} = \{A_1, \ldots, A_4\}$, and again, consider three semilattices: $S_1 = [0, \{A_2, A_3\}] \bigcup \{A_4, \{A_1, A_4\}, \mathcal{U}\}$, $S_2 = [0, \{A_2, A_3, A_4\}] \bigcup \{\mathcal{U}\}$, and $S = S_1 \bigcup S_2$. Then both L_{S_1} and L_{S_2} are in $2NF_4$, but $L_S = L_{S_1} \vee L_{S_2}$, is not.

The approximation problem was studied for so-called *choice functions* [16] (i.e. functions on sets satisfying $C(X) \subseteq X$), and it was shown that being closed under intersection/union is necessary and sufficient for the existence of a unique upper/lower approximation. This result can be easily generalized for the functions that, being ordered, form a distributive lattice (notice that choice functions ordered by $\subseteq$ form a Boolean lattice). Unfortunately, the lattice Cl_n is not close to distributive (its properties are studied in [7] and [15]), and a counterexample can be found that shows nonexistence of the unique approximation for the 2NF.

However, we can try to approximate an arbitrary scheme by that in 2NF with the same set of prime attributes. In other words, we say that a closure L' is a *2NF-approximation* of a closure L if $L' \geq L$ and $L' \in Cl_n(\mathcal{U}_p(L)) \cap 2NF_n$. (Notice that we speak of *a* 2NF-approximation).

Let us give a procedure that finds a 2NF-approximation of a given closure $L \in Cl_n$.

1. For all prime X in S_L add $[X \cap \mathcal{U}_p(L), X]$ to S_L. Denote the extended S by $\hat{S}$.
2. Extend $\hat{S}$ to a semilattice. Denote this semilattice by S'. (In other words, S' is the minimal semilattice containing $\hat{S}$.
3. Let $2NF(L) = L_{S'}$.

Proposition 1 *Given a closure L, $2NF(L)$ is a 2NF-approximation of L.*

Proof. Since $S \subseteq S', 2NF(L) \geq L$ in Cl_n. Moreover, since no new coatom appeared in S', by theorems -2 and -1 $\mathcal{U}_p(L) = \mathcal{U}_p(2NF(L))$, i.e. $2NF(L) \in Cl_n(\mathcal{U}_p(L))$. To prove that $2NF(L) \in 2NF_n$, consider an arbitrary prime X in S'. Since the families of antikeys of S and S' coincide and the antikeys unambiguously determine the keys, the keys of L and $2NF(L)$ are the same. Let $X = 2NF(L)(Y)$, where $Y \subset K$ and K is a minimal key. Then $X' = L(Y)$ is prime in S. Moreover, $X \subseteq X'$ since $L \leq 2NF(L)$. Since $X \in S', X$ is the intersection of all sets in S' that include X, X' among them. Let $X = X' \cap X_1 \cap \ldots \cap X_k$. Assume $A \in \mathcal{U}_n \cap X$. Thenm $A \in \mathcal{U}_n \cap X'$, and $X' - A \in S'$ as a set lying in $[X' \cap \mathcal{U}_p, X']$, which was added to S to get $\hat{S}$, and therefore lies in S'. Thus $X - A = (X' - A) \cap X_1 \cap \ldots \cap X_k \in S'$, which proves $[X - \mathcal{U}_n, X] = [X \cap \mathcal{U}_p, X] \subseteq S'$. Now, according to theorem 1, $2NF(L) \in 2NF_n$. □

2.2 Third Normal Form

In this section a lattice-theoretic characterization of the third normal form is given. Based on this characterization, the polynomiality of the 3NFTEST problem is proved for relations.[2] The approximation problem is solved in the case of 3NF for relations and relations schemes.

Theorem 2 *Let R be a relation over $\mathcal{U}$. Then R is in 3NF iff $[X \cap \mathcal{U}_p, X] \subseteq S_R$ for any $X \in S_R - \{\mathcal{U}\}$.*

Proof. Let R be in 3NF and $X \in S_R, X \neq \mathcal{U}$. Suppose $A \in \mathcal{U}_n$. As in the 2NF case, it is enough to show that $X - A \in S_R$. Assume $X - A$ is not closed; since X is, $L_R(X - A) = X$. Therefore $X - A \rightarrow A \in F_R$, and $X \rightarrow \mathcal{U} \in F_R$ for R is in 3NF. Closedness of X now implies $X = \mathcal{U}$, a contradiciton.

To prove the other direction, suppose $[X \cap \mathcal{U}_p, X] \subseteq S_R$ for any $X \in S_R - \{\mathcal{U}\}$; We must show that R is in 3NF. Let $X \rightarrow A \in F_R, A \in \mathcal{U}_n, A \notin X$. Let us prove that $X \rightarrow \mathcal{U} \in F_R$, or $L_R(X) = \mathcal{U}$. Suppose $L_R(X) = Y \neq \mathcal{U}$. Since $A \notin X, Y - A \in [X, Y] - \{Y\}$, and thus is not closed. But $Y - A \in [Y \cap \mathcal{U}_p, Y] \subseteq S_R$ and is therefore closed. This contradiction proves $L_R(Y) = \mathcal{U}$ and finishes the proof of the theorem. □

We denote the subset of Cl_n generated by 3NF relation schemes by $3NF_n$. Similarly to the 2NF case, this subset is not closed under the operations of Cl_n. One only has to observe that the closures L_{S_1}, L_{S_2} constructed in the previous section (for both $\vee-$ and $\wedge-$ cases) belong to $3NF_n$ while L_S does not since it is not in $2NF_n \subset 3NF_n$.

It is well-known that recognizing 3NF is $\mathcal{NP}$-complete in the case of relation schemes [14]. The situation is much better in the case of relations, where the problem has polynomial time complexity, as the following theorem shows.

Theorem 3 *There is an algorithm that, given a relation R over $\mathcal{U}$, decides whether R is in $3NF$ in a polynomial time in the number of rows and columns of R.*

Proof. Let us present an algorithm which, when given R as its input, produces a boolean variable x as the output:

1. Find the set $\mathcal{U}_p = \mathcal{U}_p(R)$.
2. Find the equality set E_R.

[2] The problem is known to be $\mathcal{NP}$-complete for relation schemes [14].

3. $x := 1$.
4. For all $X \in E_R$ and $A \in U_n = U - U_p$ such that $A \notin X$ and $X \neq U$ do the following: find the closure $L_R(X - A)$; if it equals $X - A$, go to the next pair (X, A), otherwise $x := 0$ and go to step 5.
5. Stop.

We claim that this algorithm works in polynomial time and that the output $x = 1$ iff R is in 3NF. Letus prove the first claim. Finding U_P can be done in polynomial time as shown in [8]. Constructing E_R is evidently polynomial in the size of R, and so is its own size. Finally, closure of any set can be found in $O(|E_R|)$ time, see [9]. Thus, the algorithm works in polynomial time.

Now, assume x produced by the algorithm is 0. Then for some $X \in E_R \subseteq S_R$ and $A \in U_n$ we have $X - A \notin S_R$. Then R is not in 3NF by theorem 3. Let x be 1. Let $X \in S_R, X \neq U$. Since $M(S_R) \subseteq E_R$ [9], $X = X_1 \cap \ldots \cap X_k$, where $X_1, \ldots, X_k$ are the elements of E_R which are supersets of X. Let $A \in X \cap U_n$. Since $x = 1$, $X_i - A \in S_R$ for each i. Hence $X - A = (X_1 - A) \cap \ldots \cap (X_k - A) \in S_R$ and $[X \cap U_p, X] \subseteq S_R$. Thus, R is in 3NF by theorem 3. □

Although $3NF_n$ is not closed under the operations of Cl_n, we nevertheless are able to find *the* 3NF approximation which is defined as follows:

Definition Let $L \in Cl_n$. Then $L' \in Cl_n$ is called the 3NF-*approximation* of L if the following holds:

1. $L' \geq L$;
2. $L' \in Cl_n(U_p(L)) \cap 3NF_n$ (*i.e.* $U_p(L) = U_p(L')$ and L' is in $3NF$);
3. L' is the minimal such, i.e. if $L'' \in Cl_n(U_p(L)) \cap 3NF_n$ and $L'' \geq L$, then $L'' \geq L'$.

Given a closure L, construct a closure denoted by $3NF(L)$ using the following procedure:

1. Add all intervals $[X \cap U_p, X]$ for $X \in S_L$ to S_L. Denote S_L thus extended by $\hat{S}$.
2. Extend $\hat{S}$ to the semilattice, i.e. let S' be the minimal semilattice containing $\hat{S}$.
3. $3NF(L) = L_{S'}$.

Proposition 2 *Given a closure L, $3NF(L)$ is its 3NF-approximation.*

Proof. Since $S_L \subseteq S', 3NF(L) \geq L$. According to the procedure given above, no new coatom may appear in S' and since S' is an extension of $S_L, CA(S_L) = CA(S')$. Therefore $U_p(L) = U_p(3NF(L))$ by theorems -2 and -1.

To prove $3NF(L) \in 3NF_n$, consider $X \in S', X \neq U$, and a nonprime $A \in X$. X can be represented as $X = X_1 \cap \ldots \cap X_k$, where $X_i \in [X_i^0 \cap U_p, X_i^0]$ and $X_i^0 \in S_L$. Therefore, $X_i - A \in [X_i^0 \cap U_p, X_i^0] \subseteq S'$ and $X - A = (X_1 - A) \cap \ldots (X_k - A) \in S'$. thus $3NF(L) \in 3NF_n$ by theorem 3.

If L'' is as in 3 of the definition of the 3NF-approximation, $S_L \subseteq S_{L''}$ and by theorem 3 $[X \cap U_p, X] \subseteq S_{L''}$ for any $X \in S_L, X \neq U$. Since $S_{L''}$ is a semilattice, this shows $S_{L'} \subseteq S_{L''}$ and $L'' \geq 3NF(L)$. The proposition is proved. □

Having described the 3NF-approximation, we have a natural question: how hard is it to find the approximation. Notice that the question asked is ambiguous - we did not specify what is given as an input: a relation or a relation scheme. That is, we have *two* different problems:

(3NF-APPROXIMATION FOR SCHEMES): *Given a relation scheme* $\langle U, F\rangle$, *find a scheme* $\langle U, F'\rangle$ *which is a 3NF-approximation of* $\langle U, F\rangle$ *(to put it another way,* $L_{F'} = 3NF(L_F)$*).*

(3NF-APPROXIMATION FOR RELATIONS): *Given a relation* R, *find a relation* R' *which is* R*'s 3NF-approximation (i.e.* $L_{R'} = 3NF(L_R)$*).*

The complexity result for these problems is very similar to the one for 3NFTEST - the problem is polynomial for relations and superpolynomial for schemes.

Theorem 4 *The problem 3NF-APPROXIMATION FOR RELATION can be solved in a polynomial time. The problem 3NF-APPROXIMATION FOR SCHEMES is superpolynomial provided that* $P \neq NP$.

Proof. Let R be a relation. Let $E'_R = E_R \bigcup \{X - A : X \in E_R, A \in U_n(R)\}$. Since constructing both E_R and $U_n(R)$ takes polynomial time in the size of R [8,9], E'_R can be found in polynomial time too. From theorem 3 we conclude that $M(S_L) \subseteq E'_R \subseteq S_L$, where $L = 3NF(L_R)$, and a relation R' satisfying $L_{R'} = L$ can be found by using the polynomial algorithms from [19]. This relation R' is a sought 3NF-approximation of R.

To prove the second part, show how we can use 3NF- APPROXIMATION FOR SCHEMES to solve 3NFTEST. Given a scheme $\langle U, F\rangle$, let $\langle U, F'\rangle$ be its 3NF-approximation. Notice that $\langle U, F\rangle$ is in 3NF iff $F^+ = (F')^+$. Since checking the equality $F_1^+ = F_2^+$ for two arbitrary families of FDs takes polynomial time [18], knowing F' gives rise to a polynomial algorithm that tests 3NF. Since 3NFTEST is NP-complete, $P \neq NP$ implies that approximation can not be found in a polynomial time. Note that an exponential time complexity algorithm was provided before proposition 2. The theorem is proved. □

2.3 Boyce-Codd Normal Form

In this section we discuss our main topics - characterization, testing, approximation - for BCNF. The characterization is the simplest one and corresponds to a well-known mathematical object: the order ideals. $BCNF_n$ turns out to be a sublattice of Cl_n, moreover, a distributive one. This ensures the existence of approximation, which, as in the 3NF case, can be found in polynomial time for relations and superpolynomial time for schemes.

Theorem 5 *Let* R *be a relation over* U. *Then* R *is in BCNF iff* $[\emptyset, X] \subseteq S_R$ *for any* $X \in S_R, X \neq U$.

Proof. Let R be in BCNF. Suppose $X - A \notin S_R$ for $X \in S_R - \{U\}, A \in X$. Then $X - A \rightarrow A \in F_R$, implying $X \rightarrow U \in F_R$. Thus $X - A \in S_R$ and $[\emptyset, X] \subseteq S_R$. Conversely, if the condition of the theorem holds, suppose $X \rightarrow A \in F_R, A \notin X$. If $X \rightarrow U \notin F_R$, then $L_R(X) \neq U$ and $X \in [\emptyset, L_R(X)] \subseteq S_R$, i.e. X is closed. This contradiction shows $L_R(X) = U$, i.e. R is in BCNF. □

Some similar results for BCNF were established earlier, e.g. in [20]. The following corollary gives some alternative characterizations, all of them immediately derivable from theorem 5.

Corollary 1 *Given a relation scheme* $\langle U, F\rangle$, *the following are equivalent:*

1. $\langle U, F\rangle$ *is in BCNF;*

2. $[\emptyset, X] \subseteq S_F$ *for every* $X \in S_F - \{U\}$;

3. $S_F = (\bigcup_{i=1}^{t}[\emptyset, X_i]) \bigcup \{U\}$, *where* $X_1, \ldots, X_t$ *are the antikeys;*

4. $P(U) - S_F = (\bigcup_{i=1}^{r}[K_i, U]) - \{U\}$, *where* $K_1, \ldots, K_r$ *are the minimal keys.*[3]

Let $BCNF_n$ stand for the subset of Cl_n generated y schemes in BCNF. Clearly, $BCNF_n \subseteq 3NF_n \subseteq 2NF_n$.

Proposition 3 *$BCNF_n$ is a distributive sublattice of Cl_n.*

Proof. Let $L_1, L_2 \in BCNF_n$. Since $S_{L_1} \cap S_{L_2}$ evidently satisfies the condition of theorem 5, $L_1 \wedge L_2 \in BCNF_n$. To prove $L_1 \vee L_2 \in BCNF_n$, represent S_{L_1} and S_{L_2} as $\bigcup_{i=1}^{r}[\emptyset, X_i] \bigcup \{U\}$ and $\bigcup_{i=1}^{t}[\emptyset, Y_i] \bigcup \{U\}$ respectively, see corollary 1. Let $Z_1, \ldots, Z_p$ be the maximal sets among $X_1, \ldots, X_r, Y_1, \ldots, Y_t$. Then $S_{L_1} \cup S_{L_2} \cup \{X \cap Y : X \in S_{L_1}, Y \in S_{L_2}\} = \bigcup_{i=1}^{p}[\emptyset, Z_i] \bigcup \{U\} = S_{L_1} \cup S_{L_2}$. Thus $L_1 \vee L_2 \in BCNF_n$. The sublattice $BCNF_n$ is distributive because the join and meet operations correspond to union and intersection of semilattices. □

BCNFTEST is known to be $\mathcal{NP}$-complete for relation schemes. BCNFTEST here is the problem that tests whether a subscheme of a relation scheme $\langle U, F\rangle$ generated by a proper subset $X \subset U$ is in BCNF. (Notice that checking whether the scheme itself is in BCNF takes polynomial time: one has to construct the *canonical* minimal cover [25] and check if it consists only of key dependencies). As in the 3NF case, the analogue of BCNFTEST problem for relations can be solved in polynomial time.

Proposition 4 *Given a relation R over U, BCNFTEST can be solved in polynomial time in the size of R.*

Proof. Let R be a relation and $X \subseteq U$. Let R_X denote the projection of R onto X. Denote the set of maximal elements of $E_{R_X} - \{X\}$ by E_X. Then, according to theorem 5, R_X is in BCNF iff for all $Y \in E_X$ and all $A \in Y : Y - A$ is closed, i.e. $L_{R_X}(Y - A) = Y - A$. Since constructing E_X takes polynomial time and closure can be computed in polynomial time too, the whole algorithm has polynomial time complexity. □

Similarly to th 3NF case, there exists unique approximation of a given closure by the one in BCNF. More precisely, we define *the BCNF-approximation* of a given closure L as the minimal closure L' such that $L' \geq L$ and $L' \in Cl_n(U_p(L)) \cap BCNF_n$. Let $BCNF(L)$ be the closure whose semilattice of closed sets is $\bigcup_{i=1}^{t}[\emptyset, X_i] \bigcup \{U\}$, where $X_1, \ldots, X_t$ are the antikeys of L. It follows immediately from corollary 1 and the definition of approximation:

Proposition 5 *Given a closure L, $BCNF(L)$ is its BCNF-approximation.* □

BCNF-approximation has clear interpretation in terms of FDs. If $L = L_F$ for a relation scheme $\langle U, F\rangle$, let $F' = \{K_1 \to U, \ldots, K_r \to U\}$, where $K_1, \ldots, K_r$ are the keys of $\langle U, F\rangle$. Then $L_{F'} = BCNF(L)$.

We finish this section by proving the complexity result for the approximation problem. As in the 3NF case, we have two problems:

[3]This result was proved by J. Biskup [3].

(BCNF-APPROXIMATION FOR SCHEMES): *Given a relation scheme* $\langle U, F\rangle$, *find its BCNF-approximation, i.e. a relation scheme* $\langle U, F'\rangle$ *such that* $L_{F'} = BCNF(L_F)$.

(BCNF-APPROXIMATION FOR RELATIONS): *Given a relation* R, *construct a relation* R' *which is* R*'s BCNF-approximation, i.e.* $L_{R'} = BCNF(L_R)$.

Theorem 6 *The problem BCNF-APPROXIMATION FOR RELATIONS can be solved in a polynomial time. The problem BCNF-APPROXIMATION FOR SCHEMES has exponential complexity.*

Proof. Let R be a relation and $X_1, \ldots, X_t$ its antikeys. Let $S = \bigcup_{i=1}^{t}[\emptyset, X_i] \bigcup \{U\}$. Then $L_S \in BCNF_n$ by corollary 1 and $L_S = BCNF(L_R)$ because the family of antikeys unambiguously determines the family of keys. Let $\mathcal{M}_R = \{X_i - A : A \in U, i = 1, \ldots, t\}$. Then $M(S) \subseteq \mathcal{M}_R \subseteq S$, and applying the polynomial algorithm of [19] we can construct a relation R' with $S_{R'} = S$. Since finding the antikeys takes polynomial time [8,9], R' can be constructed in a polynomial time if R is the input. Notice that $L_{R'} = L_S = BCNF(L_R)$. Thus R' is R's BCNF-approximation.

As it was mentioned before, an exponential complexity algorithm for BCNF-APPROXIMATION FOR SCHEMES exists: one has to find all minimal keys. On the other hand, it is clear that the size of the approximation F' is about the size of its canonical minimal cover [25] which consists of FDs $K_1 \to U, \ldots, K_r \to U$, where $K_1, \ldots, K_r$ are the minimal keys of $\langle U, F\rangle$ and it can be exponential: Yu and Johnson [26] have given an example of a scheme consisting of k functional dependencies on k^2 attributes with $k!$ minimal keys. For a detailed discussion of schemes reaching this extremal number of minimal keys, see [2]. □

Note that the polynomial algorithm for approximation problem for relations, described in the proof of theorem 6, was used in [13] to construct an algorithm that, given a relation, finds its minimal keys. This problem has exponenetial time complexity, but it can be decomposed into two subproblems: BCNF-APPROXIMATION, which has polynomial time complexity, and *dependency inference problem* [21] for which good practical algorithms exist.

References

[1] C. Beeri, P.A. Bernstein, Computational problems related to the design of normal form relation schemes, *ACM TODS* **4** (1979), 30-59.

[2] A. Békéssy, J. Demetrovics, Contribution to the theory of data base relations, *Discrete Matheematics* **27** (1979), 1-10.

[3] J. Biskup, private letter.

[4] G. Burosch, J. Demetrovics, G.O.H. Katona, The poset of closures as a model of changing databases, *Order* **4** (1987), 127-142.

[5] E.F. Codd, Further normalization of the data base relational model, in: *Data Base Systems*, Prentice Hall (1972), 33-64.

[6] A. Day, A lattice interpretation of database dependencies, Preprint, Lakehead University, 1989.

[7] J. Demetrovics, L. Libkin, I. Muchnik, Functional dependencies and the semilattice of closed classes, in: *Proc. of the 2nd Symp. on Math. Fund. of Database Syst.*, (J. Demetrovics, B. Thalheim eds.) *Springer LNCS* **364** (1989), 136-147.

[8] J. Demetrovics, V.D. Thi, Keys, antikeys and prime attributes, *Annales Univ. Sci. Budapest, Sect. Comp.*, **8** (1987), 35-52.

[9] J. Demetrovics, V.D. Thi, Relations and minimal keys, *Acta Cybernetica* **8** (1988), 279-285.

[10] R. Fagin, Horn clauses and database dependencies, *Journal of ACM* **29** (1982), 678-698.

[11] R. Fagin, M. Vardi, The theory of data dependencies - a survey, in: *Mathematics of Information Processing* (M. Anshel, W. Gewirtz eds.), Amer. Math. Soc. **34** (1986), 19-71.

[12] M. Gary, D. Johnson, *Computers and Intractability: A Guide to NP-completness* (W.H. Freeman and Co., San Francisco, 1979).

[13] G. Gottlob, L. Libkin, Investigations on Armstrong relations, dependency inference, and excluded functional dependencies, *Acta Cybernetica* **9** (1990), 385-402.

[14] J.H. Jou, P.C. Fischer, The complexity of recognizing 3NF relation schemes, *Inform. Process. Letters* **14** (1982), 187-190.

[15] L. Libkin, I. Muchnik, The lattice of subsemilattices of a semilattice, *Algebra Universalis*, to appear.

[16] B.M. Litvakov, Approximation of choice functions, *Automation and Remote Control* **45** (1984), 1221-1229.

[17] C.L. Lucchesi, S.L. Osborn, Candidate keys for relations, *JCSS* **17** (1978), 210-279.

[18] D. Maier, *The Theory of Relational Databases* (Computer Science Press, Rockville, MD, 1983).

[19] H. Manilla, K.-J. Räihä, Design by example: an application of Armstrong relations, *JCSS* **33** (1986), 126-141.

[20] H. Manilla, K.-J. R äihä, Practical algorithms for finding prime attributes and testing normal forms, in: *Proc. of the Symp. on Principles of Database Systems* (1989), 128-133.

[21] H. Manilla, K.-J. Dependency inference, in: *Proc. of the Conf. on Very Large Databases* (1987), 155-158.

[22] J. Paredaens, P. De Bra, M. Gyssens and D. Van Gucht, *The Structure of the Relational Datamodel* (Springer-Verlag, Berlin, 1989).

[23] B. Thalheim, *Dependencies in Relational Databases* (Stuttgart-Leipzig, 1991).

[24] J.D. Ullman, *Principles of Database Systems* (Pittman, 2nd ed., 1982).

[25] M. Wild, Implicational bases for finite closure systems, Preprint 1210, University of Darmstadt, 1989.

[26] C.T. Yu, D.T. Johnson, On the complexity of finding the set of candidate keys for a given set of functional dependencies, *Information Pocessing Lett.* **5** (1976), 100-101.

Received April 5, 1991.
Revised version received November 5, 1991.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

On Unambiguous Number Systems with a Prime Power Base

Juha Honkala*

Abstract

We study unambiguous number systems with a prime power base. Given a prime p and a p-recognizable set A, it is decidable whether or not A is representable by an unambiguous number system. Given an arbitrary integer n and n-recognizable set A, the unambiguous representation of A is unique if it exists, provided that A is not a finite union of arithmetic progressions.

Keywords: Number system, unambiguity, decidability.

1 Introduction

We study representation of integers in arbitrary number systems. Here "arbitrary" means that the digits may be larger than the base and that completeness is not required, i.e., every integer need not have a representation in the system. Also the number of the digits is arbitrary. These number systems were defined and studied by Maurer, Salomaa and Wood in [14]. The work was continued by Culik II and Salomaa in [5] and Honkala in [8]. These references discuss the connections to the theory of L systems and cryptography. Further results on number systems have been obtained in [9]-[11]. For closely related work see [1,7,13].

The study of number systems is closely connected with the study of sets of integers recognizable by finite automata. By definition, a set A of nonnegative integers is k-recognizable if and only if there exists a finite automaton which recognizes the representations of the integers of A written at base k. Here $k \geq 2$ is a positive integer. Now, if A is represented by a number system N, the representations of the integers of A can be recognized by an automaton with a single state if the digit set $\{0, 1, \ldots, k-1\}$ is replaced by the digit set of N. Thus, representability by a number system implies simplicity of recognition when the choice of the base and the digits is optimal.

In this paper we study unambiguous number systems with a prime power base. By definition, a number system is unambiguous if no integer has more than one representation in the system. The assumption concerning the base makes it possible to apply the theory of finite fields.

Suppose p is a prime. By Christol, [2], a set A of nonnegative integers is p-recognizable if and only if there exists a nonzero polynomial $P(z,t)$ over the finite field $\mathbb{F}_p$ such that $P(z, \sigma_A) = 0$. Here σ_A is the series $\sum_{i \in A} z^i$ of $\mathbb{F}_p[[z]]$. Using

*Department of Mathematics, University of Turku, 20500 Turku 50, Finland

this result we show that it is decidable whether or not a given p-recognizable set is represented by an unambiguous number system. Consequently, given a number system N with a prime power base, it is decidable whether or not there exists an unambiguous number system equivalent to N, i.e., representing the same set of integers as N.

Consider an n-recognizable set A. Here n is an arbitrary integer, not necessarily a prime. It is well known that A is also n^k-recognizable for each positive integer k. We show that, on the contrary, the set A has at most one unambiguous base, provided that A is not a finite union of arithmetic progressions. More specifically, if A is not a finite union of arithmetic progressions, there exists at most one unambiguous number system representing exactly the integers of A.

2 Definitions

By a *number system* we mean a $(v+1)$-tuple $N = (n, m_1, \ldots, m_v)$ of positive integers such that $v \geq 1, n \geq 2$ and $1 \leq m_1 < m_2 < \ldots < m_v$. The number n is referred to as the *base* and the numbers m_i as the *digits* of the number system N. A nonempty word

$$m_{i_k} m_{i_{k-1}} \ldots m_{i_1} m_{i_0}, \ 1 \leq i_j \leq v \tag{1}$$

over the alphabet $\{m_1, \ldots, m_v\}$ is said to *represent* the integer

$$m_{i_k} n^k + m_{i_{k-1}} n^{k-1} + \ldots + m_{i_1} n + m_{i_0}. \tag{2}$$

The word (1) is said to be a *representation* of the integer (2). The set of all represented integers is denoted by $S(N)$. By definition, a number system is *unambiguous* if no integer has more than one representation.

A set A of positive integers is called *representable by a number system*, shortly RNS, if there exists a number system N such that $A = S(N)$.

Suppose $k \geq 2$ and denote $\mathbf{k} = \{0, 1, \ldots, k-1\}$. Define the mapping ν_k from $\mathbf{k}^*$ to $\mathbb{N}$ by

$$\nu_k(a_0 a_1 \ldots a_m) = \sum_{i=0}^{m} a_i k^{m-i} \ (a_i \in \mathbf{k}).$$

The mapping ν_k is extended in the natural way to concern languages $L \subseteq \mathbf{k}^*$. Hence $\nu_k(L) = \{\nu_k(x) | x \in L\}$. By definition, a set A of nonnegative integers is *k-recognizable* if there exists a rational language $L \subseteq \mathbf{k}^*$ such that $A = \nu_k(L)$. For the basic properties of k-recognizable sets see [6] and [15].

The following result is essentially due to Culik II and Salomaa, [5]. For a proof, see [9].

Lemma 2.1 *If $N = (n, m_1, \ldots, m_v)$ is a number system, the set $S(N)$ is n-recognizable.*

If p is a prime, we denote by $\mathbb{F}_p$ the field of integers modulo p. The polynomial ring over $\mathbb{F}_p$ in z is denoted by $\mathbb{F}_p[z]$. The quotient field of $\mathbb{F}_p[z]$ is denoted by $\mathbb{F}_p(z)$. The ring of formal power series over $\mathbb{F}_p$ in z is denoted by $\mathbb{F}_p[[z]]$. An element α belonging to an extension field of $\mathbb{F}_p(z)$ is *algebraic* over $\mathbb{F}_p(z)$ if there exists a nonzero polynomial $P(t) \in \mathbb{F}_p(z)[t]$ such that $P(\alpha) = 0$. If α is algebraic over $\mathbb{F}_p(z)$ there exists a polynomial $R(t) \in \mathbb{F}_p(z)[t]$ of minimal degree such that $R(\alpha) = 0$ and the leading coefficient of $R(t)$ is 1. This polynomial $R(t)$ is called the

minimal polynomial of α. Notice that $R(t)$ is necessarily irreducible. By definition, the *degree* of α equals the degree of the minimal polynomial of α. The basic facts about minimal polynomials and algebraic extensions of fields which will be needed in the sequel can be found in [12].

Suppose A is a set of nonnegative integers. Then the series σ_A over $\mathbb{F}_p$ is defined by

$$\sigma_A = \sum_{i \in A} z^i.$$

The following theorem is due to Christol, [2], and Christol et al., [3].

Theorem 2.2 *Suppose that p is a prime and A is a set of nonnegative integers. The set A is p-recognizable if and only if the series*

$$\sigma_A = \sum_{i \in A} z^i$$

of $\mathbb{F}_p[[z]]$ is algebraic over the field $\mathbb{F}_p(z)$. If A is p-recognizable there exists a nonzero polynomial P in $\mathbb{F}_p[z,t]$ such that

$$P(z, \sigma_A(z)) = 0$$

and the degree of P in t is at most $p^s - 1$. Here s is the number of states in the minimal deterministic automaton recognizing the set

$$\{a_0 a_1 \dots a_h | h \geq 0, a_i \in \mathrm{p}, a_0 + a_1 p + \dots + a_h p^h \in A\}.$$

The bound given above for the degree of P can be deduced from [3, pages 407-408].

In the sequel we need a characterization of the sets A such that σ_A has degree one over $\mathbb{F}_p(z)$. By definition, a set A of nonnegative integers is *recognizable* if A is a finite union of arithmetic progressions.

Lemma 2.3 *A nonempty set $A \subseteq \mathbb{N}$ is recognizable if and only if there exists a nonzero polynomial P in $\mathbb{F}_p[z,t]$ of degree one in t such that $P(z, \sigma_A(z)) = 0$.*

Proof. If A is recognizable, the existence of P is clear. On the contrary, suppose that

$$R(z) + Q(z)\sigma_A = 0,$$

where $R(z), Q(z) \in \mathbb{F}_p[z]$. Without loss of generality we suppose that $Q(0) \neq 0$. Therefore σ_A is an $\mathbb{F}_p$-rational power series. Because $\mathbb{F}_p$ is finite, the set $A = \{i \mid \sigma_A$ contains the term $z^i\}$ is recognizable (see [16], Theorem II 5.2).

□

3 The Main Results

Suppose p is a prime. In this section we repeatedly use the following fact. If $r = \sum r_i z^i$ belongs to $\mathbb{F}_p[[z]]$ and $q \geq 1$, then

$$r^{p^q} = \sum r_i z^{ip^q}.$$

Lemma 3.1 *Suppose that $N = (n, m_1, \ldots, m_v)$ is an unambiguous number system. Suppose furthermore that $n = p^q$, where p is a prime and $q \geq 1$. Define $a(z) = \sum_{i=1}^{v} z^{m_i}$. Then $\sigma_{S(N)} \in \mathbb{F}_p[[z]]$ satisfies the equation*

$$at^n - t + a = 0$$

in $\mathbb{F}_p[[z]]$.

Proof. Clearly

$$S(N) = \bigcup_{i=1}^{v} (m_i + nS(N)) \bigcup \{m_1, \ldots, m_v\}.$$

Because N is unambiguous,

$$(m_{i_1} + nS(N)) \cap (m_{i_2} + nS(N)) = \emptyset$$

if $i_1 \neq i_2$ and $1 \leq i_1, i_2 \leq v$. Therefore

$$\begin{aligned} \sigma_{S(N)} &= a(z) + \sum_{i=1}^{v} (z^{m_i} \sum_{j \in S(N)} z^{nj}) \\ &= a(z) + a(z) \sum_{j \in S(N)} z^{nj} \\ &= a(z) + a(z)(\sum_{j \in S(N)} z^j)^n \\ &= a(z) + a(z)\sigma_{S(N)}^n. \end{aligned}$$

□

Lemma 3.2 *Suppose that $R(t) \in \mathbb{F}_p(z)[t]$ has degree $k \geq 2$ in t. Suppose furthermore that $R(t)$ divides*

$$P(t) = at^{p^q} - t + a$$

where $a \in \mathbb{F}_p[z]$, the degree m_1 of the lowest term of a is at least one and $q \geq 1$. Then

$$q \leq \log_p(m_1 k(k-1) + 1).$$

Proof. Because the derivative of $P(t)$ equals -1, the roots of $P(t)$ are simple. At least two of the roots of $P(t)$, say γ_1 and γ_2, are also roots of $R(t)$. Denote $\beta = \gamma_1 - \gamma_2$. Now

$$a\beta^{p^q} - \beta = P(\gamma_1) - P(\gamma_2) = 0. \tag{3}$$

Because β is the difference of two roots of a polynomial of degree k, the degree of the minimal polynomial of β is at most $k(k-1)$. Suppose that the minimal polynomial of β multiplied by an element of $\mathbb{F}_p[z]$ equals

$$S(t) = a_m t^m + \ldots + a_1 t + a_0,$$

where $a_i \in \mathbb{F}_p[z]$ for $0 \leq i \leq m$, and $a_m \neq 0$. Now

$$\begin{aligned} S(\beta)^{p^q} &= a_m^{p^q}(\beta^{p^q})^m + \ldots + a_1^{p^q}\beta^{p^q} + a_0^{p^q} \\ &= a_m^{p^q}(a^{-1})^m\beta^m + \ldots + a_1^{p^q}(a^{-1})\beta + a_0^{p^q}. \end{aligned}$$

Here the last equation follows by (3). Therefore, β is a root of the polynomial

$$S_1(t) = a_m^{p^q}(a^{-1})^m t^m + \ldots + a_1^{p^q}(a^{-1})t + a_0^{p^q}.$$

Because the degrees of S and S_1 are equal to the degree of the minimal polynomial of β, we have

$$a_m^{p^q-1} = a^m a_0^{p^q-1}. \tag{4}$$

Denote the degrees of the lowest terms of a_m and a_0 by i and j, respectively. Then

$$(i-j)(p^q-1) \leq m_1 k(k-1).$$

Because the degree of the lowest term of a is positive, $i > j$, and the claim follows. □

Theorem 3.3 *Suppose p is a prime. Given a p-recognizable set A, it is decidable whether or not there exists an unambiguous number system N such that $A = S(N)$.*

Proof. By Theorem 10 of [10], it is decidable whether or not A is recognizable. Suppose first it is not.

Consider the series $\sigma_A \in \mathbb{F}_p[[z]]$. By Theorem 2.2, σ_A is algebraic over $\mathbb{F}_p(z)$ and the degree of the minimal polynomial $R(t)$ of σ_A is at most $p^s - 1$. Here s is an effectively obtainable positive integer. By Lemma 2.3, the degree of $R(t)$ is at least 2. Suppose now that $A = S(N)$ where $N = (n, m_1, \ldots, m_v)$ is an unambiguous number system. By Lemma 2.1 and Cobham's theorem, [4], there exists a positive integer q such that $n = p^q$. Denote

$$a(z) = z^{m_1} + \ldots + z^{m_v}$$

and

$$P(t) = at^{p^q} - t + a.$$

By Lemma 3.1, we have $P(\sigma_A) = 0$. Therefore, because $R(t)$ is the minimal polynomial of σ_A, the polynomial $R(t)$ divides $P(t)$. By Lemma 3.2,

$$q \leq \log_p(m_1(p^s-1)(p^s-2)+1) \leq 2s + \log_p m_1.$$

Here m_1 is necessarily the least positive element of A. Therefore, to decide whether or not A is representable by an unambiguous number system, it suffices to decide whether or not $A = S(N)$ for an unambiguous number system N with a base p^i, $i \leq 2s + \log_p m_1$. This can be done by Theorem 6.3 of [5].

Suppose then that A is recognizable. The decidability in this case will be shown in the next section of this paper.

□

Corollary 3.4 *Given a number system N with a prime power base, it is decidable whether or not there exists an unambiguous number system N_1 such that $S(N) = S(N_1)$.*

The decidability status of the problems considered in Theorem 3.3 and Corollary 3.4 is open in the general case. In the last result of this section, however, no primality assumptions are needed.

Theorem 3.5 *Suppose A is a k-recognizable set for some $k \geq 2$. Furthermore, suppose that A is not recognizable. Then there is at most one unambiguous number system N such that $A = S(N)$.*

Proof. Suppose that $N_1 = (n_1, m_1, \ldots, m_u)$ and $N_2 = (n_2, m_1', \ldots, m_v')$ are distinct unambiguous number systems such that $A = S(N_1) = S(N_2)$. Because A is not recognisable there exist positive integers $n \geq 2$, i and j such that $n_1 = n^i$ and $n_2 = n^j$. Denote

$$a(z) = z^{m_1} + \ldots + z^{m_u}$$

and

$$b(z) = z^{m_1'} + \ldots + z^{m_v'}.$$

Now

$$\sigma_A(z) = a(z) + a(z)\sigma_A(z^{n^i}) \tag{5}$$

and

$$\sigma_A(z) = b(z) + b(z)\sigma_A(z^{n^j}). \tag{6}$$

Here σ_A belongs to $\mathbb{F}_p[[z]]$. The choice of the prime p is free. Replace in (5) z by z^{n^j} and in (6) z by z^{n^i}. Hence

$$\sigma_A(z^{n^j}) = a(z^{n^j}) + a(z^{n^j})\sigma_A(z^{n^{i+j}}) \tag{7}$$

and

$$\sigma_A(z^{n^i}) = b(z^{n^i}) + b(z^{n^i})\sigma_A(z^{n^{i+j}}). \tag{8}$$

Now (5)-(8) imply that

$$\sigma_A(z)[b(z)a(z^{n^j}) - a(z)b(z^{n^i})] = a(z)b(z)[a(z^{n^j}) - b(z^{n^i})]. \tag{9}$$

Because necessarily $m_1 = m_1'$, the lowest terms of $a(z)$ and $b(z)$ have the same degree. Therefore, if $n^j \neq n^i$, the right-hand side of (9) is nonzero. If $n^j = n^i$, necessarily $a(z) \neq b(z)$, and again the right-hand side of (9) is nonzero. By Lemma 2.3 this implies that A is recognisable. This contradiction proves the claim. □

Theorem 3.5 does not hold true for recognisable sets (see e.g. Example 4.4 below).

4 Representation of Recognizable Sets

In this section we give a proof of Theorem 3.3 in the case of a recognisable set.

Suppose $A \subseteq \mathbb{N}$ is recognisable. Given $k \in \mathbb{N}$, it is decidable whether or not $A \subseteq k\mathbb{N}$. Denote by d the greatest common factor of the elements of A. Because d necessarily divides the least nonzero element of A, d can be found out effectively. If $A = S(N)$ where $N = (n, m_1, \ldots, m_t)$ is a number system, d divides all the digits of N. Therefore

$$d^{-1}A = \{x | dx \in A\} = S(N')$$

where $N' = (n, d^{-1}m_1, \ldots, d^{-1}m_t)$. Clearly, N is unambiguous if and only if N' is unambiguous. Hence we suppose without loss of generality in the rest of this section that $d = 1$. Define now the ω-word $\omega(A) = a_1a_2a_3\ldots$ by

$$a_i = \begin{cases} 0 & \text{if } i \notin A \\ 1 & \text{if } i \in A. \end{cases}$$

Because A is recognizable, there exist words $u, v \in \{0,1\}^*$ such that $\omega(A) = uv^\omega$. (Here $v^\omega = vvv\ldots$). The words u and v can be obtained effectively. In the sequel we always assume that v is a primitive word, i.e., that there does not exist a word v_1 and an integer $k \geq 2$ such that $v = v_1^k$. If $i, j \geq 1$, we denote

$$\omega(A)[i, j] = a_ia_{i+1}\cdots a_{i+j-1}.$$

The length of a word w is denoted by $|w|$.

Lemma 4.1 *Suppose A is a recognizable set with $\omega(A) = uv^\omega$. If $N = (n, m_1, \ldots, m_t)$ is an unambiguous number system representing A with $n \geq |u|+|v|$, then the length of v divides n.*

Proof. Suppose that $A = S(N)$ where $N = (n, m_1, \ldots, m_t)$ is an unambiguous number system and $n \geq |u| + |v|$. Suppose m is a digit of N. Denote

$$w_1 = \omega(A)[mn + |u| + 1, |v|]$$

and

$$w_2 = \omega(A)[|u| + 1, |v|].$$

Clearly, if $m_i \in \{|u| + 1, \ldots, |u| + |v|\}$, then $mn + m_i \in \{mn + |u| + 1, \ldots, mn + |u|+|v|\}$. On the other hand, any integer in the set $\{|u|+1, \ldots, |u|+|v|\}$ belonging to A is a digit of N. Therefore the word w_1 is obtained from w_2 by replacing some 0's by 1's. However, the number of 1's in w_1 is equal to the number of 1's in w_2. Therefore $w_1 = w_2$. Because a primitive word equals no nontrivial conjugate of itself,

$$mn + |u| + 1 \equiv |u| + 1 (\bmod |v|).$$

Hence $|v|$ divides mn. Because this holds for any digit m and the greatest common factor of the digits is 1, $|v|$ divides n.

□

If $A \subseteq \mathbb{N}$, we denote $A^0 = A \cup \{0\}$. If a set B is a disjoint union of the sets $B_1, \ldots, B_s$, we denote $B = B_1 \dot\cup B_2 \dot\cup \ldots \dot\cup B_s$.

Lemma 4.2 *Suppose that A is a recognizable set of positive integers with $\omega(A) = uv^\omega$. Then there is an unambiguous number system $N = (n, m_1, \ldots, m_t)$ representing A with $n \geq |u|+|v|$ if and only if there exist a positive integer k and nonnegative integers $x_1, x_2, \ldots, x_k$ such that*

$$x_1 + A^0 \dot\cup \ldots \dot\cup x_k + A^0 = \mathbb{N} \tag{10}$$

and, furthermore, for each $x \in A$ there are infinitely many elements of A congruent to x modulo the length of v.

Proof. Suppose first that $A = S(N)$, where $N = (n, m_1, \ldots, m_t)$ is an unambiguous number system such that $n \geq |u| + |v|$. By Lemma 4.1, $|v|$ divides n. Therefore there exists an integer i with $1 \leq i \leq n$ such that

$$\{x | x \equiv i (\bmod n)\} \cap A = i + n\mathbb{N}.$$

Suppose that $y_1, \ldots, y_k$ are the digits of N that are congruent to i modulo n. Because N is unambiguous, we have

$$y_1 + nA^0 \dot{\cup} \ldots \dot{\cup} y_k + nA^0 = i + n\mathbb{N}.$$

Hence

$$\frac{1}{n}(y_1 - i) + A^0 \dot{\cup} \ldots \dot{\cup} \frac{1}{n}(y_k - i) + A^0 = \mathbb{N}.$$

It is clear that if $x \in A$ there are infinitely many elements of A congruent to x modulo the length of v.

Conversely, suppose that there exist $x_1, \ldots, x_k$ such that (10) holds. Fix $n \geq |u| + |v|$ such that $|v|$ divides n. The digits are chosen as follows. Consider an integer i with $1 \leq i \leq n$. If

$$\{x | x \equiv i (\bmod n)\} \cap A = i + n\mathbb{N},$$

then

$$i + nx_1 + nA^0 \dot{\cup} \ldots \dot{\cup} i + nx_k + nA^0 = i + n\mathbb{N} \tag{11}$$

and we take the integers $i + nx_1, \ldots, i + nx_k$ as digits. If

$$\{x | x \equiv i (\bmod n)\} \cap A = i + n + n\mathbb{N},$$

then

$$i + n + nx_1 + nA^0 \dot{\cup} \ldots \dot{\cup} i + n + nx_k + nA^0 = i + n + n\mathbb{N} \tag{12}$$

and we take $i + n + nx_1, \ldots, i + n + nx_k$ as digits. Let N be the number system that has base n and has all the digits chosen above. An inductive argument shows that $A = S(N)$. The unambiguity of N follows because we have disjoint unions in (11) and (12).

□

Notice that there exists at most one sequence $x_1, \ldots, x_k$ of nonnegative integers such that (10) holds. The existence of the sequence is easy to decide.

Suppose A is a recognizable set with $\omega(A) = uv^\omega$. Theorem 6.3 of [5] implies that all unambiguous number systems $N = (n, m_1, \ldots, m_t)$ representing A with $n < |u| + |v|$ can be constructed effectively. Lemmas 4.1 and 4.2 give all unambiguous representations with base greater than or equal to $|u| + |v|$. (This follows because no set has two distinct unambiguous representations with the same base.) In particular, given a recognizable set A, it is decidable whether or not there is an unambiguous number system N such that $A = S(N)$. This concludes the proof of Theorem 3.3.

Example 4.3 Denote $N = (2, 1, 4)$. By [5, Example 2.3], $S(N) = \{x | x \not\equiv 2 (\bmod 3), x \geq 1\}$. Clearly N is unambiguous. This example shows that sometimes a recognizable set has unambiguous bases smaller than the period. By Lemma 4.2, $S(N)$ does not have other unambiguous representations.

Example 4.4 Denote $A = \{x | x \equiv 0, 1 (\bmod 4), x \geq 1\}$. Then

$$A^0 \dot{\cup} 2 + A^0 = \mathbb{N}.$$

Therefore, by the proof of Lemma 4.2, the set A has unambiguous base $4m$ if $m \geq 1$. Hence A has infinitely many unambiguous bases. This shows that Theorem 3.5 does not hold true for recognizable sets.

References

[1] J. Berstel. Fibonacci words - a survey. In G. Rozenberg and A. Salomaa, Editors, *The Book of L,* pages 13-27, Springer-Verlag, 1986.

[2] G. Christol. Ensembles presque periodiques k-reconnaissables. *Theoret. Comput. Sci.* 9:141-145, 1979.

[3] G. Christol, T. Kamae, M. Mendes-France and G. Rauzy. Suites algebriques, automates et substitutions. *Bull. Soc. Math. de France* 108:401-419, 1980.

[4] A. Cobham. On the base-dependence of sets of numbers recognizable by finite automata. *Math. Systems Theory* 3:186-192, 1969.

[5] K. Culik II and A. Salomaa. Ambiguity and decision problems concerning number systems. *Inform. and Control* 56:139-153, 1983.

[6] S. Eilenberg. *Automata, Languages and Machines, Vol. A.* Academic Press, 1974.

[7] C. Frougny. Linear numeration systems of order two. *Inform. and Computation* 77:233-259, 1988.

[8] J. Honkala. Unique representation in number systems and L codes. *Discrete Appl. Math.* 4:229-232, 1982.

[9] J. Honkala. Bases and ambiguity of number systems. *Theoret. Comput. Sci.* 31:61-71, 1984.

[10] J. Honkala. A decision method for the recognizability of sets defined by number systems. *RAIRO Theoretical Informatics* 20:395-403, 1986.

[11] J. Honkala. On number systems with negative digits. *Annales Academiae Scientiarum Fennicae,* Series A.I. Mathematica, Vol. 14:149-156, 1989.

[12] S. Lang. *Algebra.* Addison-Wesley, 1965.

[13] A. de Luca and A. Restivo. Star-free sets of integers. *Theoret. Comput. Sci.* 43:265-275, 1986.

[14] H. Maurer, A. Salomaa and D. Wood. L codes and number systems. *Theoret. Comput. Sci.* 22:331-346, 1983.

[15] D. Perrin. Finite automata. In J. van Leeuwen, Editor, *Handbook of Theoretical Computer Science, Vol.* B, pages 1-57, North-Holland, 1990.

[16] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series.* Springer-Verlag, 1978.

Received July 1, 1991.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

Symbiotic E0L Systems

Alexander Meduna*

Summary: Cell symbiosis is described by E0L systems whose (direct) derivations are introduced on free monoids generated by finite sets of words consisting of one or two symbols. A single symbol represents a cell existing separately while two cells living symbiotically are represented by a pair of symbols. By using these systems, context sensitive and recursively enumerable languages are characterized. Thus, the presented modification remarkably increases the generative capacity of the classic concept of E0L systems.

Key words: Cell symbiosis - E0L systems - Monoids - Generative capacity

1 Introduction

1.1 Three points of view

From the *biological point of view,* this paper attempts to describe cell symbiosis (see [1]-[2]) in a simple and formal way. To do so, the classic concept of E0L systems (see [3]-[4], [11]) is modified so that their derivations are introduced on free monoids generated by finite sets of words consisting of one or two symbols. Single symbols represent cells living separately while pairs of symbols represent cells living symbiotically. Attempting to propose our formal model as universal as possible, we do not differentiate between associations of plant-plant, animal-animal, and plant-animal cells or between prokaryotic and eukaryotic cells. It is proved that our approach remarkably increases the generative capacity of E0L systems. Or, more biologically speaking, it allows us to describe some developments of cell organisms than the classical approach does not.

From the *mathematical point of view,* such a generalization of E0L systems is very natural: rather that allowing only letter monoids as domains of derivations we now introduce the derivations on free monoids generated by words consisting of one or two symbols. In other words, we investigate single finite substitutions iterated on free monoids generated by finite number of words having one or two symbols.

From the *formal language theory point of view,* the resulting grammars are very simple in comparison with some other rewriting mechanisms (see, eg, [5]-[7], [9] and references therein). Moreover, they characterize both context sensitive and recursively enumerable languages in a natural way.

*Computer Science Department, University of Missouri-Columbia, Columbia, Missouri 65211, USA and computing Centre, Technical University of Brno, CS-60200 Brno, Czechoslovakia

1.2 Relation to some other rewriting systems

Although there are some similarities between our generalization of E0L systems and EIL systems (see [11]), both of the concepts are fundamentally different: in EIL systems the way a letter is rewritten depends on its neighbors while in our systems it does not. What is restricted in our approach are derivation domains.

There are also some analogies between this paper and [8]. The latter introduces the notion of a derivation on word monoids generated by finite sets of words over total vocabularies of context free grammars. By using generators of length at most two, context sensitive and recursively enumerable languages are characterized by such modification of context-free grammars. The analogical result is proved for the generalisation of E0L systems in this paper. Thus, we get the same generative power of both parallel and sequential context-independent rewriting defined on free monoids generated by finite words having one or two symbols. Since both ways of rewriting generate quite different language families when defined on letter monoids (see [11]), this result may be of some interest.

2 Preliminaries

2.1 Basic Notions

We assume that the reader is familiar with formal language theory (see [12]), in particular, with the theory of L systems (see [11]). Some notations and definitions need perhaps an additional explanation.

For a vocabulary V, V^* denotes the letter monoid (generated by V under the operation of concatenation), e is the unit of $V^*, V^+ = V^* - \{e\}$. For a word $x \in V^*$, $|x|$ denotes the length of x. For a finite set of words W over V, W^* denotes the word monoid (generated by W under the operation of concatenation).

A context free grammar is a quadruple $G = (V, P, S, T)$, where, as usual, V is a finite alphabet, P is a finite set of productions of the form $A \longrightarrow x$, where $A \in V - T, x \in V^*, S \in V - T$ is the axiom, and $T \subseteq V$ is a terminal alphabet.

A context sensitive grammar is specified in Penttonen Normal Form $G = (V, P, S, T)$, where V, S, and T have the same meaning as for a context free grammar and every production in P is either of the form $AB \longrightarrow AC$ or $A \longrightarrow x$, where $A, B, C \in V - T, x \in (T \cup (V - T)^2)$, see Theorem 2 in [10].

A phrase structure grammar is also specified in Penttonen Normal Form $G = (V, P, S, T)$, where V, S, and T are as for a context free grammar and every production in P is either of the form $AB \longrightarrow AC$ or $A \longrightarrow x$, where $A, B, C \in V - T, x \in (\{e\} \cup T \cup (V - T)^2)$, see Theorem 4 in [10].

Given a (context free, context sensitive, or phrase structure) grammar G, in the standard manner we can introduce the relations $\Longrightarrow, \Longrightarrow^i, \Longrightarrow^+$, and $\Longrightarrow^*$ on the free monoid generated by its alphabet. If we want to express that $x \Longrightarrow y$ in G according to production p, then we write $x \Longrightarrow y\ [p]$.

2.2 Basic Definition

We now introduce a new concept of E0L systems, the subject of investigation in this paper, namely the notion of a symbiotic E0L system.

Let V be an alphabet. A *symbiotic E0L system* (SE0L-system for short) is a 4-tuple $G = (W, P, S, T)$, where $W \subseteq (V \cup V^2), P$ is a finite set of productions of

the form $A \longrightarrow x$, where $A \in V, x \in V^*, S \in V - T$, and $T \subseteq V$. G is said to be *propagating* if $A \longrightarrow x \in P$ implies $x \neq e$. G is called an *EOL system* if $V = W$.

The direct derivation relation $\Longrightarrow$ is now defined on W^* as follows: For arbitrary words $x, y \in W^*$ such that $x = a_1 a_2 \ldots a_n, a_i \in V, y = y_1 y_2 \ldots y_n, y_i \in V^*$, and productions $a_1 \longrightarrow y_1, a_2 \longrightarrow y_2, \ldots, a_n \longrightarrow y_n \in P$, we say that x *directly derives* y *according to* $a_1 \longrightarrow y_1, a_2 \longrightarrow y_2, \ldots, a_n \longrightarrow y_n$ (in G), in symbols

$$x \Longrightarrow yx \Longrightarrow y \quad [a_1 \longrightarrow y_1, a_2 \longrightarrow y_2, \ldots, a_n \longrightarrow y_n]$$

The list of applied productions (written in the brackets) is usually omitted when no confusion arises. We denote the i-fold product of $\Longrightarrow$ (for some $i \geq 0$) by $\Longrightarrow^i$, the transitive closure of $\Longrightarrow$ by $\Longrightarrow^+$, and the reflexive and transitive closure of $\Longrightarrow$ by $\Longrightarrow^*$. The *language of* G, denoted by $L(G,W)$, is defined by $L(G) = \{v \in T^* : S \Longrightarrow^* v\}$.

2.3 Denotation of Language Families

We denote by **SEOL** and **EOL** the families of languages generated by SEOL- and EOL-systems, respectively. The families of languages generated by propagating SEOL- and EOL-systems are denoted by **SEPOL** and **EPOL**, respectively. The family of context-free, context-sensitive, and recursively enumerable languages are denoted by **CF**, **CS**, and **RE**, respectively.

3 Results

3.1 Aim and Preliminary Results

In this section, we examine the generative capacity of (propagating) SEOL-systems. It follows immediately from the definitions and some basic results of formal language theory (see [11]-[12]) that

$$\mathbf{CF} \subset \mathbf{EPOL} = \mathbf{EOL} \subset \mathbf{CS} \subset \mathbf{RE}$$

and

$$\mathbf{EPOL} = \mathbf{EOL} \subseteq \mathbf{SEPOL} \subseteq \mathbf{SEOL}$$

Next we will give precision to these relationships when proving that

(i) a language is context sensitive if and only if it is generated by a propagating SEOL-system and

(ii) a language is recursively enumerable if and only if it is generated by a SEOL-system.

We close Section 3.1 by recalling a technical lemma from [8] yielding a normal form for context-sensitive grammars similar to the one given by Penttonen (see Section 2.1). We will find it useful when proving Theorem 1.

Lemma 1 *Every* $L \in \mathbf{CS}$ *can be generated by a context sensitive grammar* $G = (N_{CF} \cup N_{CS} \cup T, P, S, T)$, *where* N_{CF}, N_{CS}, *and* T *are pairwise disjoint alphabets and every production in* P *is either of the form* $AB \longrightarrow AC$, *where* $B \in N_{CS}, A, C \in N_{CF}$, *or of the form* $A \longrightarrow x$, *where* $A \in N_{CF}, x \in N_{CS} \cup T \cup N_{CF}^2$

3.2 Main Result

Now we consider the relationship between **CS** and **SEP0L**.

Theorem 1 **CS = SEP0L.**

Proof. It is straightforward to prove that **SEP0L** $\subseteq$ **CS**, hence it suffices to prove the converse inclusion.

Let L be a context sensitive language generated by a context sensitive grammar $G = (N_{CF} \cup N_{CS} \cup T, P, S, T)$ of the form described by Lemma 1.

Let

$$V = (N_{CF} \cup N_{CS} \cup T)$$

and

$$V' = V \cup Q,$$

where

$$Q = \{< A, B, C >: AB \longrightarrow AC \in P, A, C \in N_{CF}, B \in N_{CS}\}.$$

Clearly, without loss of generality, we can assume that $Q \cap V = \emptyset$.

The SEP0L-grammar, G', is defined as follows:

$$G' = (W, P', S, T),$$

where the set of productions P' is defined in the following way:

(0) for all $A \in V'$, add $A \longrightarrow A$ to P';
(1) if $A \longrightarrow x \in P, A \in N_{CF}, x \in N_{CS} \cup T \cup N_{CF}^2$, then add $A \longrightarrow x$ to P';
(2) if $AB \longrightarrow AC \in P, A, C \in N_{CF}, B \in N_{CS}$, then add the set of two productions $\{B \longrightarrow < A, B, C >, < A, B, C > \longrightarrow C\}$ to P'.

The set $W \subseteq (V \cup V^2)$ is defined as follows:

$$W = \{A < A, B, C >:< A, B, C > \in Q(A \in N_{CF})\} \cup V.$$

Obviously G is an SEP0L grammar.

Let us now introduce a function h from $(V')^*$ into V^* defined by:
for all $D \in V$, $h(D) = D$,
for all $< X, D, Z > \in Q$, $h(< X, D, Z >) = D$;
let h^{-1} be the inverse of h.

To show that $L(G) = L(G')$, we first prove two claims:

Claim 1 *If* $S \Longrightarrow^m w$ *in* $G, w \in V^+$, *for some* $m \geq 0$, *then* $S \Longrightarrow^* v$ *in* G', *where* $v \in h^{-1}(w)$.

Proof of Claim 1: This is established by induction on the length m of derivations in G.

Let $m = 0$. The only w is S because $S \Longrightarrow^0 S$ in G. Since $S \in W^*, S \Longrightarrow^0 S$ in G' and by the definition of $h^{-1}, S \in h^{-1}(S)$.

Let us suppose that our claim holds for all derivations of length at most m for some $m \geq 0$ and consider a derivation $S \Longrightarrow^{m+1} x$ in $G, x \in V^*$. Since $m + 1 \geq 1$, there is some $y \in V^+$ and $p \in P$ such that $S \Longrightarrow^m y \Longrightarrow x[p]$ in G and by the induction hypothesis there is also a derivation $S \Longrightarrow^n y'$ in G' for some $y' \in h^{-1}(y), n \geq 0$. By the definition, $y' \in W^*$.

(i) Let us first assume that $p = D \longrightarrow y_2 \in P, D \in N_{CF}, y_2 \in N_{CS} \cup T \cup N_{CF}^2, y = y_1 D y_3$, and $x = y_1 y_2 y_3, y_1 = a_1 \dots a_i, y_3 = b_1 \dots b_j$, where $a_k, b_l \in V, 1 \leq k \leq i, 1 \leq l \leq j$, for some $i, j \geq 0$ ($i = 0$ implies $y_1 = e$ and $j = 0$ implies $y_3 = e$). Since from the definition of h^{-1} it is clear that $h^{-1}(Z) = \{Z\}$ for all $Z \in N_{CF}$ we can write $y' = z_1 D z_3$, where $z_1 \in h^{-1}(y_1)$ and $z_3 \in h^{-1}(y_3)$, that is to say, $z_1 = c_1 \dots c_i, z_3 = d_1 \dots d_j$, where $c_k \in h^{-1}(a_k)$, $d_l \in h^{-1}(b_l)$, for $1 \leq k \leq i, 1 \leq l \leq j$. It is clear that $D \longrightarrow y_2 \in P'$, see (1).

Let $d_1 \notin Q$. Then, it is easy to see that $z_1 y_2 z_3 \in W^*$ and so

$$z_1 D z_3 \Longrightarrow z_1 y_2 z_3 \; [c_1 \longrightarrow c_1, \dots, c_i \longrightarrow c_i, D \longrightarrow y_2, d_1 \longrightarrow d_1, \dots, d_j \longrightarrow d_j]$$

in G'. Therefore, $S \Longrightarrow^n z_1 D z_3 \Longrightarrow z_1 y_2 z_3$ and $z_1 y_2 z_3 \in h^{-1}(y_1 y_2 y_3)$.

Let $d_1 \in Q$, that is, $Dh(d_1) \longrightarrow DC \in P$ (for some $C \in N_{CF}$), see the definition of h. Hence, we have $h(d_1) \longrightarrow d_1$ in P', see (2) (observe that this production is the only production in P' that has d_1 appearing on its right-hand side). It is clear, by the definition of W, that $d_2 \notin Q$. Thus, $\{z_1 Dh(d_1) d_2 \dots d_j, z_1 y_2 h(d_1) d_2 \dots d_j\} \subseteq W^*$. Since $S \Longrightarrow^n z_1 D d_1 \dots d_j$ in G', there must exist the following derivation in G':

$$\begin{aligned} S &\Longrightarrow^{n-1} z_1 Dh(d_1) d_2 \dots d_j \\ &\Longrightarrow \quad z_1 D d_1 d_2 \dots d_j \quad [c_1 \longrightarrow c_1, \dots, c_i \longrightarrow c_i, D \longrightarrow D, \\ &\qquad\qquad\qquad\qquad h(d_1) \longrightarrow d_1, d_2 \longrightarrow d_2, \dots, d_j \longrightarrow d_j] \end{aligned}$$

in G'. So, we get

$$\begin{aligned} S &\Longrightarrow^{n-1} z_1 Dh(d_1) d_2 \dots d_j \\ &\Longrightarrow \quad z_1 y_2 h(d_1) d_2 \dots d_j \quad [c_1 \longrightarrow c_1, \dots, c_i \longrightarrow c_i, D \longrightarrow y_2, \\ &\qquad\qquad\qquad\qquad h(d_1) \longrightarrow h(d_1), d_2 \longrightarrow d_2, \dots, d_j \longrightarrow d_j] \end{aligned}$$

such that $z_1 y_2 h(d_1) d_2 \dots d_j$ is in $h^{-1}(x)$.

(ii) Let $p = AB \longrightarrow AC \in P, A, C \in N_{CF}, B \in N_{CS}, y = y_1 AB y_2, y_1, y_2 \in V^*, x = y_1 AC y_2, y' = z_1 AY z_2, z_i \in h^{-1}(y_i), i \in 1, 2, Y \in h^{-1}(B)$, and $y_1 = a_1 \dots a_i, y_3 = b_1 \dots b_j, a_k, b_l \in V, 1 \leq k \leq i, 1 \leq l \leq j$, for some $i, j \geq 0$. Let $z_1 = c_1 \dots c_i, z_3 = d_1 \dots d_j, c_k \in h^{-1}(a_k), d_l \in h^{-1}(b_l), 1 \leq k \leq i, 1 \leq l \leq j$. Clearly, $\{B \longrightarrow < A, B, C >, < A, B, C > \longrightarrow C\} \subseteq P'$, see (2), and $A < A, B, C > \in W$, see the definition of W.

Let $Y = B$. Since $y' \in W^*$ and $B \in N_{CS}, d_1 \notin Q$. Consequently, $z_1 A < A, B, C > z_2$ and $z_1 AC z_2$ are in W^* by the definition of W. Thus,

$$\begin{aligned} S &\Longrightarrow^n z_1 AB z_2 \\ &\Longrightarrow \quad z_1 A < A, B, C > z_2 \quad [c_1 \longrightarrow c_1, \dots, c_i \longrightarrow c_i, A \longrightarrow A, \\ &\qquad\qquad\qquad\qquad B \longrightarrow < A, B, C >, d_1 \longrightarrow d_1, \dots, d_j \longrightarrow d_j] \end{aligned}$$

$$\begin{aligned} &\Longrightarrow z_1 AC z_2 \; [c_1 \longrightarrow c_1, \dots, c_i \longrightarrow c_i, A \longrightarrow A, \\ &\qquad\qquad < A, B, C > \longrightarrow C, d_1 \longrightarrow d_1, \dots, d_j \longrightarrow d_j] \end{aligned}$$

and $z_1 AC z_2 \in h^{-1}(x)$.

Let $Y \in Q$. Clearly, $h(Y)$ must be equal to B. By (2) and the definition of Q, we have $B \longrightarrow Y \in P'$. Clearly, $z_1 AC z_2$ is in W^* for $d_1 \notin Q$ as we have already shown. Thus, since $S \Longrightarrow^n z_1 AY z_2$ in G', the word $z_1 AY z_2$ can be derived in G' as follows:

$$S \Longrightarrow^{n-1} z_1ABz_2$$
$$\Longrightarrow \quad z_1AYz_2 \quad [c_1 \longrightarrow c_1, \ldots, c_i \longrightarrow c_j, A \longrightarrow A,$$
$$B \longrightarrow Y, d_1 \longrightarrow d_1, \ldots, d_j \longrightarrow d_j].$$

Since $z_1A < A, B, C > z_2$ and z_1ACz_2 belong to W^*, we get

$$S \Longrightarrow^{n-1} z_1ABz_2$$
$$\Longrightarrow \quad z_1A < A, B, C > z_2 \quad [c_1 \longrightarrow c_1, \ldots, c_i \longrightarrow c_i, A \longrightarrow A,$$
$$B \longrightarrow < A, B, C >, d_1 \longrightarrow d_1, \ldots, d_j \longrightarrow d_j]$$

$$\Longrightarrow z_1ACz_2 \; [c_1 \longrightarrow c_1, \ldots, c_i \longrightarrow c_i, A \longrightarrow A,$$
$$< A, B, C > \longrightarrow C, d_1 \longrightarrow d_1, \ldots, d_j \longrightarrow d_j]$$

in G', where $z_1ACz_2 \in h^{-1}(x)$.

The points (i) and (ii) cover all possible rewriting of y in G. Thus, the claim now follows by the principle of induction.

Claim 2 *If $S \Longrightarrow^n u$ in G', $u \in W^*$, for some $m \geq 0$, then $S \Longrightarrow^* h(u)$ in G.*

Proof of Claim 2: This is established by induction on the length m of derivations.

For $n = 0$ the only u is S because $S \Longrightarrow^0 S$ in G'. Since $S = h(S)$ we have $S \Longrightarrow^0 S$ in G.

Let us assume the claim holds for all derivations of length at most n, for some $n \geq 0$, and consider a derivation $S \Longrightarrow^{n+1} u$, where $u \in W^*$. Since $n+1 \geq 1$, there is some $v \in W^*$, such that $S \Longrightarrow^n v \Longrightarrow u[p]$ in G' and by the induction hypothesis $S \Longrightarrow^* h(v)$ in G.

We will first prove the following statement (*):

Let $v = rDs$ and $p = D \longrightarrow z \in P$ in G'. Then $h(v) \Longrightarrow^i h(r)h(z)h(s)$ in G, for some $i = 0, 1$. (*)

To verify (*), consider the following three cases:

(i) Let $h(z) = h(D)$, see (2). Then $h(v) \Longrightarrow^0 h(r)h(z)h(s)$ in G.

(ii) Let $z \in (T \cup N_{CS} \cup N_{CF}^2), D \in N_{CF}$ Then there is a production $B \longrightarrow z \in P$, see (1), and by the definition of h we have $B \longrightarrow z = h(B) \longrightarrow h(z)$. Thus, $h(r)h(D)h(s) \Longrightarrow h(r)h(z)h(s)[h(B) \longrightarrow h(z)]$ in G.

(iii) Let $z = C \in N_{CF}$ and $D =< A, B, C >$ for some $< A, B, C > \in Q$, see (2). By the definition of W, we have $r = tA$, where $t \in W^*$ and so $v = tACs$. By the definition of Q, there is a production $AB \longrightarrow AC \in P$. Thus, $tABs \Longrightarrow tACs[AB \longrightarrow AC]$ in G where $tABs = h(tA)h(< A, B, C >)h(s)$ and $tACs = h(tA)h(C)h(s)$.

By inspection of P', the points (i) through (iii) cover all possible types of productions in P', proving (*).

It should be clear that by using (i) through (iii) we can construct the derivation $h(v) \Longrightarrow^i h(u)$, for some $i \in \{0, \ldots, |u|\}$, in the following way: first we rewrite all occurrences of symbols corresponding to the case (iii) and then all occurrences of symbols corresponding to (ii); the technical details are left to the reader.

Thus, $S \Longrightarrow^n h(v) \Longrightarrow^i h(u)$ in G. Hence, by the principle of induction, we have established Claim 2.

Now, the proof of the equivalence of G and G' can be derived from Claim 1 and 2:

By the definition of h^{-1}, we have $h^{-1}(a) = \{a\}$ for all $a \in T$. Thus, by Claim 1, we have for any $x \in T^*$:

$$S \Longrightarrow^* x \text{ in } G \text{ implies } S \Longrightarrow^* x \text{ in } G',$$

that is, $L(G) \subseteq L(G')$.

Conversely, since $T^* \subseteq W^*$, we get, by the definition of h and Claim 2, for any $x \in T^*$:

$$S \Longrightarrow^* x \text{ in } G' \text{ implies } S \Longrightarrow^* x \text{ in } G,$$

that is, $L(G') \subseteq L(G)$.

Thus, $L(G) = L(G')$ and so **CS = SEP0L**, which proves the theorem.

□

3.3 Some Corollaries an Conclusion

First of all, Theorem 1 and the definitions yield the following normal form:

Corollary 1 *Let L be a context sensitive language over an alphabet T. Then L can be generated by an SEP0L system $G = (W, T, P, S)$, where W is over an alphabet V such that $T \subseteq W, (W - V) \subseteq (V - T)^2$, and if $A \longrightarrow x$ and $1 < |x|$ then $x \in (V - T)^2$.*

Let us now turn to the investigation of SE0L-grammars with erasing productions. We will show that these grammars generate precisely the family of recursively enumerable languages.

Corollary 2 **RE = SE0L.**

Proof. Clearly we have the containment **SE0L** $\subseteq$ **RE**, hence it suffices to show **RE** $\subseteq$ **SE0L.**

Each language $L \in$ **RE** can be generated by a phrase structure grammar G in Penttonen Normal Form (see Section 2.1) which can be converted to the grammar of an analogical form to the one described by Lemma 1 (except that the former may contain some erasing productions), see [8]. Thus, the containment **RE** $\subseteq$ **SE0L** can be proved by analogy with the techniques used in the proof of Theorem 1. The details are left to the reader.

□

Since the forms of the resulting SE0L-grammar in the proof of Corollary 2 and that in the proof of Theorem 1 are analogical, we get the following:

Corollary 3 *Let L be a recursively-enumerable language over an alphabet T. Then L can be generated by an SE0L system $G = (W, T, P, S)$, where W is over an alphabet V such that $T \subseteq W, (W - V) \subseteq (V - T)^2$, and if $A \longrightarrow x$ and $1 < |x|$ then $x \in (V - T)^2$.*

Finally, summing up the main results of this paper, we obtain:

Corollary 4 **SEP0L = CS $\subset$ SE0L = RE.**

Acknowledgement. The author is indebted to Anton Novacky for useful discussions during preparation of this paper.

References

[1] Henry, S. M.: Symbiosis, Vol. 1. Academic Press, New York, 1966.

[2] Margulis, L.: Symbiosis in Cell Evolution. W.H. Freeman and Company, San Francisco, 1981.

[3] Meduna, A.: A Note on Exponential Density of ETOL Languages. Kybernetika, Vol. 22, pp. 514-518 (1986)

[4] Meduna, A.: ETOL Languages are Exponentially Dense. Proceeding of the Conference: Application of Artificial Intelligence, pp. 287-295, Prague (1987)

[5] Meduna, A.: Evaluated Grammars. Acta Cybernetika, Vol. 8, pp. 169-176 (1987)

[6] Meduna, A.: Characterization of the Chomsky Hierarchy through Sequential-Parallel Grammars, Rostock. Math. Kolloq., Vol. 32, pp. 4-14 (1987)

[7] Meduna, A. and Horvath, Gy.: On State Grammars. Acta Cybernetica, Vol. 8, pp. 237-245 (1988)

[8] Meduna, A.: Context Free Derivations on Word Monoids. Acta Informatica, Vol. 27, pp. 781-786 (1990)

[9] Meduna, A.: Generalized Forbidding Grammars. International Journal of Computer Mathematics, Vol. 36, No. 4, pp. 31-38 (1990)

[10] Penttonen, M.: One-Sided and Two-Sided Context in Formal Grammars. Inform. Contr. 25, 371-392 (1974)

[11] Rozenberg, G. and Salomaa, A.: The Mathematical Theory of L Systems. Academic Press, London, 1980.

[12] Salomaa, A.: Formal Languages. Academic Press, London, 1973.

Received August 17, 1990.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

Alternation bounds for tree automata

Kai Salomaa *

Abstract

We consider alternation depth bounds for tree automata, that is, we limit the number of alternations of existential and universal computation steps. We show that a constant bound guarantees that the forest recognized is regular, whereas already a logarithmic bound enables the automata to recognize a strictly larger class of forests. As a corollary we obtain results on other types of alternation bounds for tree automata. We consider also commutation properties of independent computation steps of an alternating tree automaton.

1 Introduction

An alternating computation generalizes a nondeterministic one by allowing the automaton to both make existential choices and branch the computation universally. Alternation has been used to model parallelism of various machine models, cf. e.g. [2], [6], [8], [9], [10]. The alternating computations of tree automata are particularly interesting because there parallelism occurs on two levels: the automaton reads in parallel independent subtrees of the input and, furthermore, the computation can branch universally.

It is well known that alternating finite automata recognize only the regular languages. Also alternating top-down finite tree automata recognize just the regular forests, cf. [18]. On the other hand, alternation increases dramatically the computational power of finite bottom-up tree automata, cf. [14], [15], [17]. These automata define as yield-languages even all recursively enumerable languages. Other alternating tree automaton models are studied in [11], [12], [16].

Because of the great computational power of alternating tree automata it seems natural to consider restrictions on the alternating computations that would limit the family of recognized forests. Alternation bounds on various Turing machine models and multihead finite automata have been investigated in [3], [5], [6], [7], [10], [13]. There one restricts the width or leaf-size of the computation trees, (or equivalently the number of universal computation steps in an alternating computation.) Also bounds that restrict the size of the alternating computation trees have been investigated, this corresponds to limiting together the length and parallelism of the computation.

Alternation depth of Turing machines, that is the number of alternations of existential and universal configurations in a computation path, is considered in [2]. A similar measure called alternation size which restricts the total number of alternations in a computation tree is defined in [1], and the alternation size of a

*Department of Mathematics, University of Turku, SF-20500 Turku, Finland

Turing machine is shown to correspond closely to the number of reversals of an auxiliary pushdown automaton.

Here we study alternation depth bounds in the context of bottom-up tree automata. For tree automata, the configurations cannot be divided into existential and universal ones as in the case of Turing machines (a tree configuration can contain an arbitrary number of states) and, therefore, we will in fact limit the alternation of existential and universal computation steps in any branch of the computation tree. Intuitively, one may think the edges of the computation tree to be labeled by E or U depending on whether the computation step is existential or universal. On a path from the root to a leaf a sequence of consequtive symbols E (respectively U) that is limited on both sides by symbols U (resp. E) is said to be an existential (resp. universal) computation segment. In an alternation bounded computation tree the number of computation segments on each path is limited by a function on the size of the input.

Clearly even a computation tree with a constant alternation bound may contain an arbitrary number of universal computation steps and thus its width can be linear in the size of the input. However, it is shown that forests recognized by tree automata with a constant alternation bound are regular. This illustrates the fact that the power of alternating tree automata is not only due to an unlimited number of parallel computations but also to the capability of alternating existential and universal computation steps and reading independent subtrees of the input differently in distinct branches of the computation. As a special case it follows that also automata with constant width computation trees define only the regular forests.

On the other hand, it is shown that already a logarithmic alternation bound enables the finite tree automata to recognize also nonregular forests. The main open question is whether the same is true for some sublogarithmic (nonconstant) functions.

In order to establish the above results we need to consider the commutation properties of independent computation steps in an alternating computation. (That is, computation steps at independent nodes of the input tree.) In general, the order in which the recognizer reads independent subtrees of the input can be essential. However, clearly two existential (i.e., nondeterministic) independent computation steps always commute, and the same is true for universal computation steps. Furthermore, a universal and existential computation step semi-commute, that is, a universal computation step may be assumed to be performed first.

In Section 2 we recall the definition of an alternating tree recognizer and establish the above commutation properties. Alternation bounded computations are defined in Section 3 and there it is also shown that constant bounds define only the regular forests. Sections 4 and 5 consider the logarithmic bound and some related alternation measures.

2 Alternating tree automata

The reader is assumed to be familiar with trees and tree automata, cf. e.g. [4]. Here we recall the definition of an alternating finite bottom-up tree recognizer, cf. [14], [15], [17]. Since we will consider computations with a bounded number of alternations of existential and universal computation steps, we use a slightly different definition from that of *op. cit.* There at each computation step the automaton was able to make both an existential and a universal choice. It is shown that in terms of recognition power these models are essentially equivalent.

The set of (nonempty) words over a set X is denoted by X^* (respectively X^+)

and the empty word by λ. The length of a word w is denoted $|w|$. The catenation of $A, B \subseteq X^*$ is defined by $AB = \{w \in X^* | (\exists w_1 \in A)(\exists w_2 \in B) w = w_1 w_2\}$. The set of subsets of X is denoted $\wp(X)$. If X is finite, the cardinality of X is denoted by $\#X$. If X is a (proper) subset of Y this is denoted by $X \subseteq Y (X \subset Y)$. The symbol N_+ stands for the set of positive integers.

A tree domain D is a nonempty subset of N_+^* that satisfies the conditions (TD1) and (TD2) below.

(TD1) If $u \in D$, then every prefix of u belongs to D.

(TD2) For every $u \in D$ there exists $i \in N_+ \cup \{0\}$ such that $uj \in D$ iff $1 \leq j \leq i$. ($i = 0$ if u has no daughters.)

Let A be a set. An A-labeled tree is a mapping $t : D \to A$, where D is a tree domain. The elements of D are called nodes of the tree t and D is denoted dom(t). A node u is said to be labeled by $t(u) (\in A)$. We use freely notions such as the height, root, a leaf, and a subtree of a tree. The height of t is denoted hg(t) (a tree with one node is defined to have height zero), and the set of leaves of t is denoted leaf(t). The subtree of t at node u is denoted t/u. A node v is a successor of $u \in$ dom(t) if u is a prefix of v, and u and v are said to be independent, $u \| v$, if neither one is a successor of the other. If $u_1, \ldots, u_m \in$ dom(t) are pairwise independent, then $t(u_1 \leftarrow t_1, \ldots, u_m \leftarrow t_m)$ denotes the tree obtained from t by replacing t/u_i with $t_i, i = 1, \ldots, m$.

In the term notation one assumes that a node with i daughters (immediate successors) is always labeled by a symbol of rank i. Letters Σ and Ω denote here finite ranked alphabets and the set of m-ary, $m \geq 0$, symbols of Σ is denoted by Σ_m. The rank of an element $\sigma \in \Sigma_m$ is denoted $\text{rank}_\Sigma(\sigma)$ or just $\text{rank}(\sigma)$ if the alphabet Σ is known. Let A be a (finite) set. The set of ΣA-trees, $F_\Sigma(A)$, is the smallest set such that (i) $\Sigma_0 \cup A \subseteq F_\Sigma(A)$, and (ii) if $\sigma \in \Sigma_m, m \geq 1, t_1, \ldots, t_m \in F_\Sigma(A)$, then $\sigma(t_1, \ldots, t_m) \in F_\Sigma(A)$. The set of Σ-trees, F_Σ, is defined to be $F_\Sigma(\emptyset)$. Subsets of F_Σ are called Σ-forests. Let $t, t_1, \ldots, t_m \in F_\Sigma(A)$ and $a_1, \ldots, a_m \in A$. Then $t(a_1 \leftarrow t_1, \ldots, a_m \leftarrow t_m)$ denotes the ΣA-tree obtained from t by replacing every occurrence of a symbol a_i with t_i. To a given Σ-tree t one associates in the natural way the corresponding tree domain dom(t). For $t \in F_\Sigma$ we denote size(t) = #dom(t). A (Σ-)tree t is said to be balanced if every path from the root of t to a leaf has equal length.

We still recall the notion of tree homomorphism, cf. [4]. For every $m \geq 1$, let $\Xi_m = \{\xi_1, \ldots, \xi_m\}$ be a set of variables. Assume that for every $m \geq 0$ such that $\Sigma_m \neq \emptyset$ we are given a mapping $h_m : \Sigma_m \to F_\Omega(\Xi_m)$. Then the mappings h_m determine inductively a tree homomorphism $h : F_\Sigma(A) \to F_\Omega(A)$ as follows.

(i) $h(a) = a$ for $a \in A$.

(ii) Let $m \geq 0, \sigma \in \Sigma_m, t_1, \ldots, t_m \in F_\Sigma(A)$. Then

$$h(\sigma(t_1, \ldots, t_m)) = h_m(\sigma)(\xi_1 \leftarrow h(t_1), \ldots, \xi_m \leftarrow h(t_m)).$$

It is clear that Σ-trees can be seen as Σ-labeled trees (graphs) where each node having i daughters is labeled by an element of rank i and conversely every Σ-labeled tree with the above property corresponds to a unique Σ-tree. In the following, we speak about trees using interchangeably the above notions of a Σ-labeled graph and an element of F_Σ.

Definition 2.1 *An* **alternating (bottom-up) tree recognizer** *is a four-tuple* $\mathbb{A} = (\Sigma, A, A', G)$, *where*

(i) Σ *is a finite ranked alphabet,*

(ii) A *is a finite set of states,*

(iii) $A' \subseteq A$ *is the a set of final states, and*

(iv) G is the state transition relation defined as follows. The relation G associates with every $\sigma \in \Sigma_m, m \geq 0$, a mapping

$$\sigma_G : A^m \to \wp(A) \times \{E, U\}.$$

The class of alternating tree recognizers is denoted by **ATR**. Let **A** be as in Definition 2.1. Elements of $F_\Sigma(A)$ are called **A-configurations.** A **configuration tree of A** is a tree where the nodes are labeled **A**-configurations, and the set of configuration trees of **A** is denoted by CT(**A**). Let $T \in$ CT (**A**). Then conf(T) denotes the set of all **A**-configurations labeling some node of T.

Let K be an **A**-configuration. Subtrees of K of the form $\sigma(a_1, \ldots, a_m)$, $m \geq 0$, $\sigma \in \Sigma_m, a_1, \ldots, a_m \in A$ are said to be **active subtrees**, and the set of (occurrences of) active subtrees is denoted by act(K). Let $Z \in \{E, U\}$ and assume that

$$\sigma_G(a_1, \ldots, a_m) = (B, Z), \tag{1}$$

where $\sigma \in \Sigma_m, a_1, \ldots, a_m \in A$ and $B \subseteq A$. Then we denote

$$\sigma_{Z(G)}(a_1, \ldots, a_m) = B,$$

or simply,

$$\sigma_Z(a_1, \ldots, a_m) = B$$

if G is known. We also denote $G = E(G) \cup U(G)$ or simply $G = E \cup U$. This notation can be justified by the fact that always exactly one of the sets $\sigma_E(a_1, \ldots, a_m)$ and $\sigma_U(a_1, \ldots, a_m)$ is defined.

If in (1) $Z = E$, we say that the active subtree $f = \sigma(a_1, \ldots, a_m)$ is **existential** (or **of type E**), and if $Z = U$, f is said to be **universal** (or **of type U**). Also $\sigma_Z(a_1, \ldots, a_m)$ is denoted simply by f_Z. If f_Z consists of only one element, we say informally that the corresponding computation step is deterministic. When considering specific examples, the state-transition relation G is usually convenient to define by listing all nonempty sets f_Z where f is an active subtree.

Intuitively, $\sigma_Z(a_1, \ldots, a_m)$ denotes the set of next states the automaton **A** reaches after reading the input symbol σ in states $a_1, \ldots, a_m$. If $\sigma(a_1, \ldots, a_m)$ is existential, the automaton chooses nondeterministically one of the next states in which it continues the computation. If $Z = U$, the computation has to be continued in all states of $\sigma_U(a_1, \ldots, a_m)$. This is defined formally below.

Definition 2.2 *The transition relation $\Rightarrow_{\mathbf{A}}$ of a recognizer $\mathbf{A} \in ATR$ is the binary relation on* CT(**A**) *defined as follows. Let $T_1, T_2 \in$ CT(**A**). Then $T_1 \Rightarrow_{\mathbf{A}} T_2$ if T_2 is obtained from T_1 as follows. Let n be a leaf of T_1 that is labeled by an **A**-configuration K. Let $f \in act(K)$ be of type $Z, Z \in \{E, U\}$, and $f_Z = \{a_1, \ldots, a_m\}, a_i \in A, i = 1, \ldots, m$. If $Z = E$, then T_2 is obtained from T_1 by attaching for the node n a daughter labeled by $K(f \leftarrow a_i)$ for some $i \in \{1, \ldots, m\}$. If $Z = U$, then in T_2 the node n has m daughters labeled by the configurations $K(f \leftarrow a_1), \ldots, K(f \leftarrow a_m)$.*

Let $K \in F_\Sigma(A)$. The set of K-computation trees of **A** is defined by

$$\text{COM}(\mathbf{A}, K) = \{T \in \text{CT}(\mathbf{A}) | K \Rightarrow_{\mathbf{A}}^* T\}.$$

Above K denotes the configuration tree with one node labeled by K. A computation tree is **accepting** if all its leaves are labeled by elements of A', and the set of

accepting K-computation trees is denoted $\mathrm{ACOM}(\mathbf{A}, K)$. We say that $\mathbf{A}$ accepts a configuration K if $\mathrm{ACOM}(\mathbf{A}, K) \neq \emptyset$. The set of accepted $\mathbf{A}$-configurations is denoted by $\mathrm{ACC}(\mathbf{A})$. The forest recognized by the recognizer $\mathbf{A}$ is

$$L(\mathbf{A}) = F_\Sigma \cap \mathrm{ACC}(\mathbf{A}).$$

The family of forests recognized by alternating tree recognizers is denoted $\mathbf{L(ATR)}$.

Let $T_1, T_2 \in \mathrm{COM}(\mathbf{A}, t)$ for some input t and $T_1 \Rightarrow_{\mathbf{A}} T_2$. We use the notation $T_1 \Rightarrow_{\mathbf{A}}{}^{E} T_2$ (resp. $T_1 \Rightarrow_{\mathbf{A}}{}^{U} T_2$) to indicate that T_2 is obtained from T_1 using an existential (resp. universal) computation step in some configuration labeling a leaf of T_1. A computation tree T is said to be **complete** if each leaf of T is labeled by an element of A (i.e., each branch of the computation of T has reached the root of t.)

Example 2.3 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR$ *where* $\Sigma = \Sigma_0 \cup \Sigma_2, \Sigma_0 = \{\tau, \gamma\}, \Sigma_2 = \{\omega\}, A = \{a, b\}, A' = \{a\}$ *and the relation* $G = E(G) \cup U(G)$ *is defined by the following:*

$$\tau_{U(G)} = \{a, b\}, \gamma_{E(G)} = \{a, b\}, \omega_{E(G)}(a, a) = \omega_{E(G)}(b, b) = \{a\},$$

$$\omega_{U(G)}(a, b) = \omega_{U(G)}(b, a) = \{b\}.$$

Let $t = \omega(\tau, \gamma)$. We construct an accepting t-computation tree of $\mathbf{A}$. Below we denote a computation tree T with leaves labeled by configurations $K_1, \ldots, K_m$, $m \geq 1$, simply by $[K_1, \ldots, K_m]$.

$$\begin{aligned} t \;&\Rightarrow_{\mathbf{A}}{}^{U}\; [\omega(a,\gamma), \omega(b,\gamma)] \Rightarrow_{\mathbf{A}}{}^{E} [\omega(a,a), \omega(b,\gamma)] \\ &\Rightarrow_{\mathbf{A}}{}^{E}\; [\omega(a,a), \omega(b,b)] \Longrightarrow_{\mathbf{A}}{}^{E} [a, \omega(b,b)] \Longrightarrow_{\mathbf{A}}{}^{E} [a, a]. \end{aligned}$$

It is easy to see that this is the only accepting t-computation tree of $\mathbf{A}$, i.e., in an accepting computation on t, $\mathbf{A}$ must always read the leaf labeled by τ before the leaf γ. (Of course, the computation steps in different branches of the computation tree can be performed in arbitrary order.)

Let $\mathbf{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$. The recognizer $\mathbf{A}$ is said to be **nondeterministic** if all active subtrees of $\mathbf{A}$ are existential. If, furthermore, for all active subtrees f the set f_E contains at most one element, the recognizer $\mathbf{A}$ is said to be **deterministic**. If $\mathbf{A}$ is deterministic and $f_E = \{b\}, b \in A$, we denote f_E simply by b. (A deterministic recognizer could of course also be defined as a special case of a universal one.) It is clear that this definition of nondeterministic and deterministic tree recognizers is equivalent to the usual definitions, cf. [4]. Nondeterministic and determinisitic bottom-up recognizers both recognize the family of regular forests **REG**.

It is known from [14], [15], [17] that alternating bottom-up tree recognizers recognise also nonregular forests. In order to simplify some constructions, in *op. cit.* one uses an automaton model that in each computation step can branch the computation both existentially and universally. In the following we refer to this model as the **generalized alternating automaton** (or recognizer). The definition used here is more restricted as each computation step has to be purely existential or universal. The automaton model of Definition 2.1 cannot straightforwardly simulate the generalised alternating recognizers, however the models are "essentially equivalent" in the sense described below.

Let Σ be a ranked alphabet and M a Σ-forest recognised by a generalized alternating tree automaton **B**. Define the ranked alphabet $\Omega = \Sigma \cup \Lambda$ where $\Lambda = \{\sigma' | \sigma \in \Sigma\}$ and $\Omega_m = \Sigma_m$ if $m \neq 1$ and $\Omega_1 = \Sigma_1 \cup \Lambda$. Let $h : F_\Sigma \to F_\Omega$ be the tree homomorphism defined by

$$h_m(\sigma) = \sigma'(\sigma(\xi_1, \ldots, \xi_m)), m \geq 0, \sigma \in \Sigma_m.$$

Intuitively for $t \in F_\Sigma$, $h(t)$ is obtained simply by attaching above each node labeled by a symbol σ a node labeled with the unary symbol σ'. We claim that an automaton **A** of Definition 2.1 can recognise the Ω-forest $h(M)$. The proof of this result is essentially the same as the proof of Theorem 4.4 of [14] and we just explain it intuitively below. At a node labeled by σ, **A** performs the existential choice of the generalized alternating automaton **B** recognising M, and then at the node above labeled by σ' the recognizer **A** simulates the universal choice of **B**. Note that by Lemma 2.6 below, without restriction one can assume that **A** performs the universal computation step in σ' immediately after reading the symbol σ, (this could also be seen directly as in the proof given in [14].) Thus it is clear that all computations of **A** on a tree $h(t), t \in F_\Sigma$, correspond to a computation of the generalized automaton on t, and $L(\mathbf{A}) = L(\mathbf{B})$.

Intuitively, the above result means that corresponding to every forest L recognised by a generalised alternating automaton, the family $L(\mathrm{ATR})$ contains a forest essentially similar to L. Now by the results of [14], [15], [17] it is immediate that $L(\mathrm{ATR})$ contains forests that are not regular, and even not context-free (that is, algebraic). Also for instance the emptiness and equivalence problems are undecidable for $L(\mathrm{ATR})$. The tree homomorphism h above does not affect the yield (or frontier), cf. [4], of a forest and hence by [17] it follows that every recursively enumerable language is the yield of a forest in $L(\mathrm{ATR})$.

Finally we define a complete recognizer. An ATR-recognizer is said to be complete if $f_Z \neq \emptyset$ for all active subtrees f of type $Z, Z \in \{E, U\}$. The proof of the following lemma is then immediate.

Lemma 2.4 *Every forest of L(ATR) can be recognized by a complete ATR-recognizer.*

Example 2.5 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR$ *where* $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2, \Sigma_0 = \{\tau, \gamma\}, \Sigma_1 = \{\sigma\}, \Sigma_2 = \{\omega\}, A = \{a, b, b_1, b_2, c, c_1, c_2, d, d_1, d_2, e, e_1, e_2, f\}, A' = \{f\}$ *and the relation* $G = E \cup U$ *is defined by the following:*

(i) $\tau_U = \{a, b\}$
(ii) $\gamma_E = \{d_1, d_2, e\}$
(iii) $\sigma_E(a) = \{c\}$
(iv) $\sigma_U(c) = \{c_1, c_2\}$
(v) $\sigma_E(b) = \{b_1, b_2\}$
(vi) $\sigma_U(b_i) = \{b_i\}, i = 1, 2$
(vii) $\sigma_U(d_i) = \{d_i\}, i = 1, 2$
(viii) $\sigma_U(e) = \{e_1, e_2\}$
(ix) $\omega_E(x, y) = \{f\}$ *if* $(x, y) = (c_i, d_i)$ *or* $(x, y) = (b_i, e_i), i \in \{1, 2\}$.

The relation G is undefined in all cases not covered by (i)-(ix). We denote $t = \omega(\sigma\sigma(\tau), \sigma(\gamma))$ and construct an accepting t-computation tree of **A**. As in the previous example we denote a computation tree with the sequence of configurations

labeling its leaves:

$$
\begin{aligned}
t \;&\Rightarrow_{\mathbf{A}}{}^{U}[\omega(\sigma\sigma(a),\sigma(\gamma)),\omega(\sigma\sigma(b),\sigma(\gamma))]\\
&(\Rightarrow_{\mathbf{A}}{}^{E})^{2}[\omega(\sigma(c),\sigma(\gamma)),\omega(\sigma\sigma(b),\sigma(e))]\\
&(\Rightarrow_{\mathbf{A}}{}^{U})^{2}[\omega(c_1,\sigma(\gamma)),\omega(c_2,\sigma(\gamma)),\omega(\sigma\sigma(b),e_1),\omega(\sigma\sigma(b),e_2)]\\
&(\Rightarrow_{\mathbf{A}}{}^{E})^{4}[\omega(c_1,\sigma(d_1)),\omega(c_2,\sigma(d_2)),\omega(\sigma(b_1),e_1),\omega(\sigma(b_2),e_2)]\\
&(\Rightarrow_{\mathbf{A}}{}^{U})^{4}[\omega(c_1,d_1),\omega(c_2,d_2),\omega(b_1,e_1),\omega(b_2,e_2)]\\
&(\Rightarrow_{\mathbf{A}}{}^{E})^{4}[f,f,f,f]
\end{aligned}
$$

It can be verified that the computation tree constructed above is the unique accepting t-computation tree of $\mathbf{A}$. Of course the computation steps in parallel branches of the computation tree can be performed in different order, but the resulting computation tree will always be unique (if it is accepting.) Also one sees that in the computation starting from the configuration $\omega(\sigma\sigma(a),\sigma(\gamma))$ one must read both σ-symbols in the subtree $\sigma\sigma(a)$ before the symbol γ whereas in the computation starting from $\omega(\sigma\sigma(b),\sigma(\gamma))$ the automaton necessarily has to read the subtree $\sigma(\gamma)$ before continuing the computation in the subtree $\sigma\sigma(b)$.

This example illustrates the fact that in an alternating computation the order in which subtrees of the input are processed can be very essential. This is the reason why an alternating computation cannot in general be simulated by a nondeterministic one using a subset construction. The above example was constructed to be as simple as possible and here of course $L(\mathbf{A})$ is regular, (in fact $L(\mathbf{A}) = \{t\}$.) For examples of alternating tree recognizers defining nonregular forests see [14], [15], [17].

In Examples 2.3 and 2.5 we noticed that the order in which independent subtrees of the input are read can be important. To conclude this section we investigate when computation steps in independent subtrees commute.

As was done in the previous examples, it is many times convenient to denote a computation tree T with the sequence of configurations $[K_1,\ldots,K_m]$ labeling the leaves of T. The configurations $K_i, i = 1,\ldots,m$, contain all information needed to continue the computation of T and also their order is irrelevant. We say that computation trees T_1 and T_2 are equivalent if the leaves of both T_1 and T_2 are labeled by the same sequence of configurations. In this case T_1 can be completed to an accepting computation tree iff the same holds for T_2. In general the computation tree gives also the structure of the computation and we cannot for all purposes replace it with the sequence of its leaves.

An arbitrary sequence of $\mathbf{A}$-configurations $[K_1,\ldots,K_m]$ does not necessarily correspond to any computation tree but we can extend the relation $\Rightarrow_{\mathbf{A}}$ in the natural way for arbitrary sequences of configurations:

$$[K_1,\ldots,K_m] \Rightarrow_{\mathbf{A}} [H_1,\ldots,H_n]$$

iff for some $i \in \{1,\ldots,m\}$ there exists a computation tree T with leaves labeled by $M_1,\ldots,M_r$ such that $K_i \Rightarrow_{\mathbf{A}} T$ and the multiset $\{K_1,\ldots,K_{i-1},M_1,\ldots,M_r,K_{i+1},\ldots,K_m\}$ equals to the multiset $\{H_1,\ldots,H_n\}$. (Here T is a tree of height one with the root labeled by K_i.) Note in particular that if $[K_1,\ldots,K_m]$ is the sequence of leaves of a computation tree T_1 and $[H_1,\ldots,H_n]$ are the leaves of T_2 then $[K_1,\ldots,K_m]\Rightarrow_{\mathbf{A}}{}^{*}[H_1,\ldots,H_n]$ holds if $T_1\Rightarrow_{\mathbf{A}}{}^{*}T_2$.

In the next lemma we prove commutation properties of alternating computations. There it is notationally more convenient to consider sequences of con-

figurations instead of computation trees. Using the above definition the alternating computations can be defined also for sequences that do not correspond to any computation tree and we thereby prove a slightly more general result. We still introduce some notation. Let $\mathbf{A} = (\Sigma, A, A', G) \in ATR$. Let $Z \in \{E, U\}$ and $K_i, H_j \in F_\Sigma(A), i = 1, \ldots, r, j = 1, \ldots, s$. Then the notation $[K_1, \ldots, K_r] \Rightarrow_{\mathbf{A}}{}^{Z}(u, i)[H_1, \ldots, H_s], i \in \{1, \ldots, r\}, u \in \mathrm{dom}(K_i)$, is used to denote that the sequence of configurations $[H_1, \ldots, H_s]$ is obtained from $[K_1, \ldots, K_r]$ by applying a Z-computation step at node u in the configuration K_i. If $r = 1$ or i is otherwise clear we denote $\Rightarrow_{\mathbf{A}}{}^{Z}(u, i)$ simply by $\Rightarrow_{\mathbf{A}}{}^{Z}(u)$.

Lemma 2.6 *Let $\mathbf{A} = (\Sigma, A, A', G) \in ATR$ and $K \in F_\Sigma(A), u, v \in \mathrm{dom}(K)$. Assume that $u \| v$ and $K/u, K/v \in \mathrm{act}(K)$. Denote $K/u = f, K/v = g$, and let $f_X = \{a_1, \ldots, a_m\}, g_Y = \{b_1, \ldots, b_n\}, m, n \geq 1, X, Y \in \{E, U\}$.*

(i) Assume that f and g are existential, (i.e., $X = Y = E$.) Let

$$[K] \Rightarrow_{\mathbf{A}}{}^{E}(u)[K_1] \Rightarrow_{\mathbf{A}}{}^{E}(v)[K_2].$$

Then there exists $K' \in F_\Sigma(A)$ such that

$$[K] \Rightarrow_{\mathbf{A}}{}^{E}(v)[K'] \Rightarrow_{\mathbf{A}}{}^{E}(u)[K_2].$$

(ii) Assume that f is existential and g universal. Let

$$[K] \Rightarrow_{\mathbf{A}}{}^{E}(u)[K_1] \Rightarrow_{\mathbf{A}}{}^{U}(v)[H_1, \ldots, H_n].$$

Then there exist configurations $H_i', i = 1, \ldots, n$, such that

$$[K] \Rightarrow_{\mathbf{A}}{}^{U}(v)[H_1', \ldots, H_n'](\Rightarrow_{\mathbf{A}}{}^{E}(u))^n[H_1, \ldots, H_n].$$

(iii) Assume that f and g are universal. Suppose that

$$\begin{aligned} &[K] \Rightarrow_{\mathbf{A}}{}^{U}(u)[K_1, \ldots, K_m] \Rightarrow_{\mathbf{A}}{}^{U}(v, i) \\ (*)\quad &[K_1, \ldots, K_{i-1}, K_i(v \leftarrow b_1), \ldots, K_i(v \leftarrow b_n), K_{i+1}, \ldots, K_m] \Rightarrow_{\mathbf{A}}{}^{*}[M_1, \ldots, M_r], \end{aligned}$$

where $v \notin \mathrm{dom}(M_j)$ or $M_j(v) \in A, 1 \leq j \leq r$, i.e., in each of the configurations M_j the active subtree g has been read.

Then there exists a computation

$$\begin{aligned} &[K] \Rightarrow_{\mathbf{A}}{}^{U}(v)[K(v \leftarrow b_1), \ldots, K(v \leftarrow b_n)] \Rightarrow_{\mathbf{A}}{}^{U}(u, 1) \\ &[K_1(v \leftarrow b_1), \ldots, K_m(v \leftarrow b_1), K(v \leftarrow b_2), \ldots, K(v \leftarrow b_n)](\Rightarrow_{\mathbf{A}}{}^{U}(u))^{n-1} \\ (**)\quad &[K_1(v \leftarrow b_1), \ldots, K_m(v \leftarrow b_1), \ldots, K_1(v \leftarrow b_n), \ldots, K_m(v \leftarrow b_n)] \\ &\Rightarrow_{\mathbf{A}}{}^{*}[M_1, \ldots, M_r]. \end{aligned}$$

Proof. (i) This is immediate since $K_1 = K(u \leftarrow a_i)$ and $K_2 = K(u \leftarrow a_i, v \leftarrow b_j), i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$, and clearly we can choose $K' = K(v \leftarrow b_j)$.
(ii) Now $K_1 = K(u \leftarrow a_i), i \in \{1, \ldots, m\}$, and $H_j = K(u \leftarrow a_i, v \leftarrow b_j), j = 1, \ldots, n$. Thus we have

$$\begin{aligned} [K] \quad &\Rightarrow_{\mathbf{A}}{}^{U}(v)[K(v \leftarrow b_1), \ldots, K(v \leftarrow b_n)] \\ &\Rightarrow_{\mathbf{A}}{}^{E}(u, 1)[K(u \leftarrow a_i, v \leftarrow b_1), K(v \leftarrow b_2), \ldots, K(v \leftarrow b_n)] \\ &\Rightarrow_{\mathbf{A}}{}^{E}(u, 2) \ldots \Rightarrow_{\mathbf{A}}{}^{E}(u, n)[H_1, \ldots, H_n]. \end{aligned}$$

(iii) Clearly $K_i = K(u \leftarrow a_i), i = 1, \ldots, m$. Let $N_{1j}, \ldots, N_{sj}$ be the configurations from the sequence $M_1, \ldots, M_r$ that are successors of $K_j, j \in \{1, \ldots, i-1, i+1, \ldots, m\}$. Since the configurations $K_i(v \leftarrow b_1), \ldots, K_i(v \leftarrow b_n)$ appear in both (*) and (**), it is sufficient to show that

$$[K_j(v \leftarrow b_1), \ldots K_j(v \leftarrow b_n)] \Rightarrow_{\mathbf{A}}{}^*[N_{1j}, \ldots, N_{sj}], j \in \{1, \ldots, m\}, j \neq i. \quad (2)$$

Let $j \in \{1, \ldots, i-1, i+1, \ldots, m\}$. Let $H_1, \ldots, H_q$ be the configurations appearing in each branch of the computation $[K_j] \Rightarrow_{\mathbf{A}}{}^*[N_{1j}, \ldots, N_{sj}]$ just before the automaton reads the subtree g at node v. Thus $[K_j] \Rightarrow_{\mathbf{A}}{}^*[H_1, \ldots, H_q]$ using only computation steps at nodes that are independent with v. From this it follows that

$$[K_j(v \leftarrow b_r)] \Rightarrow_{\mathbf{A}}{}^*[H_1(v \leftarrow b_r), \ldots, H_q(v \leftarrow b_r)], r = 1, \ldots, n. \quad (3)$$

By the choice of $H_1, \ldots, H_q$,

$$[H_1(v \leftarrow b_1), \ldots, H_1(v \leftarrow b_n), \ldots, H_q(v \leftarrow b_1), \ldots, H_q(v \leftarrow b_n)]$$

$$\Rightarrow_{\mathbf{A}}{}^*[N_{1j}, \ldots, N_{sj}]$$

and hence (2) holds by (3). (Note that one may arbitrarily permute the configurations $P_1, \ldots P_z$ in a sequence $[P_1, \ldots, P_z]$.) Q. E. D.

In the above lemma, case (i) states that one may always permute two independent (i.e., corresponding to independent nodes) existential computation steps and (ii) states that an existential computation step followed by a universal computation step may be replaced by first making the universal step and thereafter the corresponding existential computation steps. Case (iii) states that always two independent universal computation steps commute. This is the most complicated case as one does not directly obtain identical configurations but has to consider the computation so far that in each branch both universal active subtrees have been read. This is not a restriction when considering accepting computation trees where each branch ends at the root of the input.

The fourth case would be a universal computation step followed by an independent existential computation step. These cannot (in general) be permuted since in each universal branch the automaton can make different existential choices. Thus one can say that independent existential and universal computation steps semi-commute: one may always replace EU with UE but not in the other direction.

3 Depth bounded alternation

As observed in the previous section, alternation is a very powerful mode of computation for bottom-up tree automata. For this reason we consider alternating computations where in each path the number of alternations of the existential and universal computation steps is bounded by some function on the size of the input tree.

Definition 3.1 *We define a mapping alt:* $\{E, U\}^+ \rightarrow N_+$ *as follows. Let* $u \in \{E, U\}^+$. *Then* $alt(u)$ *is the least integer* n *such that we can write*

$$u = u_1 \ldots u_n, \; u_i \in E^+ \cup U^+, i = 1, \ldots, n.$$

Definition 3.2 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR, t \in F_\Sigma$ *and* $T \in COM(\mathbf{A}, t)$. *We define a mapping* $\phi_T : dom(T) \to \{E, U\}^*$ *inductively as follows.*

(i) $\phi_T(\lambda) = \lambda$.

(ii) Let $u \in dom(T)$ *be labeled by an* $\mathbf{A}$*-configuration* K. *Suppose that in the computation of* T *in the configuration* K *the recognizer reads an active subtree of type* $Z, Z \in \{E, U\}$. *Let* $n = \max\{i | ui \in dom(T)\}$. *Then for every* $j = 1, \ldots, n$:

$$\phi_T(uj) = \phi_T(u)Z.$$

(Note that $n \leq 1$ *if* $Z = E$.*)*

Now we define the function alt: $COM(\mathbf{A}, t) \to N_+$ *by*

$$\text{alt}\,(T) = \max\{\text{alt}\,(\phi_T(u)) | u \in \text{leaf}\,(T)\}.$$

Let u be a node of a computation tree T. Then $\phi_T(u)$ gives the sequence in which existential and universal computation steps are performed along the path from the root of T to u. Thus $\text{alt}(\phi_T(u))$ denotes the number of existential and universal computation segments in the computation corresponding to the node u. Now depth bounded alternating computations can be defined by restricting the value $\text{alt}(T)$.

Definition 3.3 *Let* $\mathbf{A} = (\Sigma, A, A', G) \in ATR, t \in F_\Sigma$, *and* $\theta : N_+ \to N_+$ *be a function. The set of* θ**-bounded t-computation trees of A** *is*

$$\text{COM}(\mathbf{A}, t)[\theta] = \{T \in \text{COM}(\mathbf{A}, t) | \text{alt}(T) \leq \theta(\text{size}(t))\}.$$

A forest $L \subseteq F_\Sigma$ *is* θ**-bounded recognized by A if**

(i) $L = L(\mathbf{A})$, *and*

(ii) for every $t \in L$, $COM(\mathbf{A}, t)[\theta] \cap ACOM(\mathbf{A}, t) \neq \emptyset$.

In this case we denote $L = L(\mathbf{A})[\theta]$. *The family of forests* θ*-bounded recognized by alternating tree recognizers is denoted* $L(ATR)[\theta]$.

Thus L is θ-bounded recognised by $\mathbf{A}$ if each tree t of L has an accepting computation tree with alternation depth at most $\theta(\text{size}(t))$ and any tree not in L does not have an accepting computation. If $L(\mathbf{A})[\theta]$ is defined, we say also that the recognizer $\mathbf{A}$ is θ**-bounded**. Note that if one would define $L(\mathbf{A})[\theta]$ just to consist of trees $t \in F_\Sigma$ such that there exists an accepting computation tree in $\text{COM}(\mathbf{A}, t)[\theta]$, then the automaton would be able to use the counting properties of the function θ to check properties of the inputs. This would clearly be unnatural, especially if the function θ is not well behaved. Note that $L(\mathbf{A})[\theta]$ is not defined if $L(\mathbf{A})$ contains trees that cannot be accepted in θ-bounded computations.

Let t and $\mathbf{A}$ be as in Example 2.5 and let T be the t-computation tree considered there. Then $\text{alt}(T) = 6$. Thus $L(\mathbf{A})[\theta] = \{t\}$ for every function θ such that $\theta(6) \geq 6$. (Note that $\text{size}(t) = 6$.)

Lemma 3.4 *For every function* θ, *the family* $L(ATR)[\theta]$ *is closed with respect to intersection with regular sets.*

Proof. This is seen easily by adding to the states of an ATR-recognizer second components that simulate the computation of a deterministic recognizer for the regular forest in question. Clearly the simulation can be done at the same time preserving the type (existential or universal) of each computation step. Q.E.D.

Theorem 3.5 *Suppose that $\theta_1(n) \leq \theta_2(n)$ almost everywhere, i.e., there exists $M \in N_+$ such that for all $n \geq M, \theta_1(n) \leq \theta_2(n)$. Then*

$$L(ATR)[\theta_1] \subseteq L(ATR)[\theta_2].$$

Proof. Let L be a Σ-forest θ_1-bounded recognized by $\mathbb{A} \in \mathrm{ATR}$. Denote $F(\geq) = \{t \in F_\Sigma \mid \mathrm{size}(t) \geq M\}$ and $F(<) = \{t \in F_\Sigma \mid \mathrm{size}(t) < M\}$. Now

$$L = (L(\mathbb{A})[\theta_1] \cap F(\geq)) \cup (L(\mathbb{A})[\theta_1] \cap F(<)).$$

By Lemma 3.4 there exists $\mathbb{B} \in \mathrm{ATR}$ such that $L(\mathbb{B})[\theta_1] = L(\mathbb{A})[\theta_1] \cap F(\geq)$. By the choice of M, furthermore, $L(\mathbb{B})[\theta_1] = L(\mathbb{B})[\theta_2]$. Since $F(<)$ is finite, using $\mathbb{B}$ one can easily construct a recognizer $\mathbb{B}'$ such that

$$L(\mathbb{B}')[\theta_2] = L(\mathbb{B})[\theta_2] \cup (L(\mathbb{A})[\theta_1] \cap F(<)) = L.$$

Q.E.D.

Clearly $L(\mathrm{ATR})[id] = L(\mathrm{ATR})$ where id denotes the identity function. In the following we will consider constant and logarithmic alternation bounds. Let $c(k), k \geq 1$, denote the function that maps every element of N_+ to k. Then $L(\mathrm{ATR})[c(1)] = \mathrm{REG}$ because both the purely existential and purely universal tree automata recognize only the regular forests. Next we will show that in fact $L(\mathrm{ATR})[c(k)] = \mathrm{REG}$ for all $k \geq 1$. In the following we denote the function $c(k)$ simply by k.

A first idea for a regularity proof for the forests of $L(\mathrm{ATR})[k]$ might be to simulate the k-bounded alternating computations by a deterministic tree recognizer using a subset construction where the sets would additionally contain the information which existential (resp. universal) segment of the computation one is simulating. (There can be at most k segments.) However, this approach does not work because it may be the case that in different branches of the computation a given node must be read in different segments.

To illustrate the difficulty, let us consider again from Example 2.5 the t-computation tree which is the unique accepting computation tree for the input $t = \omega(\sigma\sigma(\tau), \sigma(\gamma))$. For instance, in the left branch of the computation the symbol γ has to be read in the second existential segment whereas in the right branch it is necessarily read in the first existential segment.

It turns out that a deterministic automaton simulating the computations of $\mathbb{A}$ will need to store in the states the information concerning the partition into existential and universal segments of all possible computation trees of the input scanned so far, this will be called the *computation schema*. It will be seen that for k-bounded computations the number of distinct computation schemata is finite. Let $\mathbb{A} \in \mathrm{ATR}$, $t = \sigma(t_1, \ldots, t_m) \in F_\Sigma$ and T be a complete t-computation tree of $\mathbb{A}$. The computation tree T is obtained by combining t_i-computation trees, $i = 1, \ldots, m$, and finally in each branch reading the root symbol σ. Thus it is clear that one can construct an arbitrary (k-bounded) t-computation tree if one knows all possible (k-bounded) t_i-computation trees.

In the following $\mathbb{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$ is always assumed to be complete. By Lemma 2.4 this is not a restriction. (Clearly the analogy of Lemma 2.4 holds also for arbitrary θ-bounded computations.) We say that $K \in F_\Sigma(A)$ is an existential configuration if all active subtrees of K are existential and otherwise K is said to be universal. In particular, if $K \in A$ then $\mathrm{act}(K) = \emptyset$ and hence K is existential.

Lemma 3.6 *Let K be a universal configuration. Then there exist unique existential configurations $K_1, \ldots, K_n$ such that*

$$[K](\Rightarrow_{\mathbb{A}}{}^{U})^*[K_1, \ldots, K_n]. \tag{4}$$

(Of course $K_1, \ldots, K_n$ may be arbitrarily permuted.)

Proof. Since **A** is complete, there are existential configurations $K_1, \ldots, K_n$ such that (4) holds (one computes universal active subtrees until there are none left.) By Lemma 2.6 (iii) universal computation steps commute and hence the configurations $K_1, \ldots, K_n$ are unique.

Definition 3.7 *Let $t \in F_\Sigma$ and $k \geq 1$. The* **k-bounded t-computation schema of A**, *$SC(t, k, \mathbf{A})$ is the configuration tree S defined as follows. The root of S is labeled by t. Suppose that a node $u \in dom(S)$, $|u| \leq k-1$, is labeled by a configuration K.*

(i) Suppose that K is universal and let $K_1, \ldots, K_n$ be the (by Lemma 3.6 unique) existential configurations such that $[K](\Rightarrow_{\mathbf{A}}{}^{U})^*[K_1, \ldots, K_n]$. Then the node u has n daughters (immediate successors) labeled by $K_1, \ldots, K_n$.

(ii) Let K be existential, $K \notin A$. Denote by C the set of all configurations K' such that $[K](\Rightarrow_{\mathbf{A}}{}^{E})^*[K']$ and K' is universal or $K' \in A$. Then for every $K' \in C$ the node u has a daughter labeled by K'.

(iii) If $K \in A$ then u is a leaf of S.

Finally, if $u \in dom(S)$ and $|u| = k$, then u has no daughters.

If u is a leaf of S and $S(u) \notin A$, then this branch corresponds to a computation that does not reach the root of the tree t in k existential and universal computation segments. These computations cannot be a part of any k-bounded computation on an input with subtree t and thus the corresponding branches can be pruned from the schema.

The **pruned schema**, $pr(S)$, is obtained from S by recursively repeating the following. Choose a leaf u of S labeled by an element not belonging to A and let v be the mother (immediate predecessor) of u. If the configuration $S(v)$ is existential then remove the node u. If $S(v)$ is universal then remove all daughters of v. (If a universal configuration K has a daughter leading to failure then the computation has failed already in K.) Note that $pr(S)$ is the empty tree iff there does not exist a complete k-bounded t-computation tree of **A**. In this case t is not a subtree of any tree of $L(\mathbf{A})|k|$.

The pruned computation schema $pr(\mathrm{SC}(t, k, \mathbf{A}))$ will be denoted by $\mathrm{PSC}(t, k, \mathbf{A})$ and in the following, when not otherwise mentioned, by a computation schema we always mean the pruned schema.

Suppose that $\mathrm{S} = \mathrm{PSC}(t, k, \mathbf{A})$. An **A**-configuration tree T is said to be a **configuration tree associated with the schema S** if T is constructed as follows. The root of T is labeled by t. Suppose that $u \in \mathrm{dom}(S) - \mathrm{leaf}(S)$ is labeled by a configuration K.

(i) If K is universal, then the node u has in T all the same daughters as in S.

(ii) If K is existential, then u has exactly one daughter labeled by some configuration that is a daughter of K in S.

Thus a configuration tree associated with the schema S is essentially a t-computation tree where some intermediate nodes in the existential and universal computation segments have been removed.

Example 3.8 *Let $\mathbf{A} = (\Sigma, A, A', G)$ be the recognizer from Example 2.3 and $t = \omega(\omega(\tau, \gamma), \gamma)$. Then the 2-bounded t-computation schema $SC(t, 2, \mathbf{A})$ is given in Figure 1. The configurations $a, \omega(\omega(a, \gamma), \gamma)$, and $\omega(\omega(b, \gamma), \gamma)$ are existential, all other configurations appearing in the schema are universal. The pruned computation schema $PSC(t, 2, \mathbf{A})$ is obtained by removing all leaves except the ones labeled by a. The schema $PSC(t, 2, \mathbf{A})$ has only one associated configuration tree and it equals to $PSC(t, 2, \mathbf{A})$.*

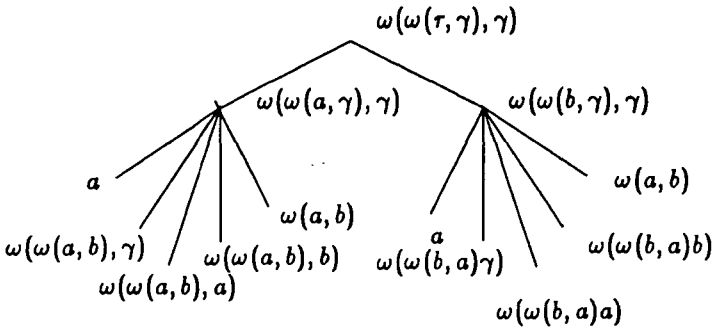

Figure 1.

It is clear that every configuration tree associated with a schema corresponds to a computation tree of the recognizer **A**. In fact we have the correspondence also in the converse direction.

Let $\mathbf{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$ and $t \in F_\Sigma, k \geq 1$. We say that $T \in \mathrm{COM}(\mathbf{A}, t)$ is normalized if the following condition holds. If some leaf of t is universal (i.e., t is a universal configuration), then at the root of T the recognizer reads a universal active subtree.

Lemma 3.9 *Let* **A** *and* t *be as above and* $k \geq 1$. *Then there exists a complete normalized computation tree in* $COM(\mathbf{A}, t)[k]$ *with leaves labeled by* $a_1, \ldots, a_n$ $(a_i \in A)$ *iff there is a configuration tree associated with the schema* $PSC(t, k, \mathbf{A})$ *having leaves* $a_1, \ldots, a_n$.

Proof. The proof in the "if"-direction is immediate since a configuration tree T associated to the schema is clearly a computation tree of **A** where some intermediate nodes are left out. According to the definition of $\mathrm{PSC}(t, k, \mathbf{A})$ if t has a universal leaf-symbol, then the computation of T first branches universally, i.e., the corresponding computation tree is normalized.

Suppose then that $T \in \mathrm{COM}(\mathbf{A}, t)[k]$ is normalized and has leaves $a_1, \ldots, a_n$. Using the commutation properties of Lemma 2.6 we show that there exists an equivalent $T_1 \in \mathrm{COM}(\mathbf{A}, t)[k]$ (i.e., also T_1 has the leaves $a_1, \ldots, a_n$) that follows the computation in the schema $\mathrm{PSC}(t, k, \mathbf{A})$.

Since T is normalized, if t contains a universal leaf-symbol the automaton first makes a universal computation step. By Lemma 2.6 universal computation steps commute with each other and semi-commute with existential computation steps. Thus in an equivalent computation tree one can first make all possible universal computation steps (in arbitrary order). Now the computation of T_1 begins as in the schema $\mathrm{PSC}(t, k, \mathbf{A})$. Always in an existential configuration K the automaton makes an arbitrary number of consequtive existential computation steps that lead to some universal configuration K'. Thus K' is a daughter of K in $\mathrm{PSC}(t, k, \mathbf{A})$. In K', **A** makes a universal computation step and thus again by Lemma 2.6 it can be made to read all universal active subtrees of K' successively yielding a number of existential configurations. By Lemma 3.6, these are exactly the daughters of

K' in $\mathrm{PSC}(t,k,\mathbf{A})$. (Note that if a configuration K contains a universal active subtree f then by Lemma 2.6 in a K-computation tree $\mathbf{A}$ could always first read f. In general this could cause additional alternations of the existential and universal computation steps. However here this problem does not occur because the original computation step made in K' is universal and thus one can make thereafter an arbitrary number of universal steps "for free".)

Since T is k-bounded so is also T_1 (any operations above do not increase the existential-universal alternations.) Thus T is equivalent to a configuration tree associated with the schema $\mathrm{PSC}(t,k,\mathbf{A})$. Q.E.D.

The previous lemma gives almost a criterion for checking whether $t \in L(\mathbf{A})[k]$ using only the schema $\mathrm{PSC}(t,k,\mathbf{A})$. There is still the restriction that the computation tree has to be normalized. This restriction can be removed by considering $(k+1)$-bounded schemata.

Lemma 3.10 *Let $\mathbf{A} = (\Sigma, A, A', G) \in ATR$, $k \geq 1$, and $L = L(\mathbf{A})[k]$. Then $t \in L$ iff there exists a configuration tree W associated with the schema $PSC(t, k+1, \mathbf{A})$ such that all leaves of W are labeled by elements of A'.*

Proof. Let $t \in L$ and $T \in \mathrm{COM}(\mathbf{A}, t)[k]$ be accepting. Assume that at least one leaf of t is labeled by a universal symbol. Then, by Lemma 2.6, T can be transformed to an equivalent computation tree T_1 where first the automaton performs all possible universal computation steps, i.e., T_1 is normalized. Furthermore, from the proof of Lemma 2.6 it follows that T_1 is $(k+1)$-bounded. Moving a number of universal computation steps to the beginning may introduce an additional universal computation segment if the computation of T starts existentially. (Of course it is also possible that $\mathrm{alt}(T_1) < k$, but for our purposes it is sufficient just to know the upper bound $\mathrm{alt}(T_1) \leq k+1$.) On the other hand, if all leaves of t are existential then already the computation tree T is normalized. Thus in both cases by Lemma 3.9 there exists a configuration tree W associated with the schema $\mathrm{PSC}(t, k+1, \mathbf{A})$ such that the leaves of W are labeled by elements of A' (since T_1 is accepting).

Conversely assume that W as above exists. Then by Lemma 3.9, there exists an accepting computation tree in $\mathrm{COM}(\mathbf{A}, t)[k+1]$. Thus $t \in L(\mathbf{A})$ and there necessarily exists also an accepting k-bounded t-computation tree. Q.E.D.

According to Lemmas 3.9 and 3.10 the schema $\mathrm{PSC}(t, k+1, \mathbf{A})$ contains the information on all complete k-bounded t-computation trees of $\mathbf{A}$. We want to define a deterministic tree automaton that stores the schemata in its states. For this purpose we need to consider the composition of schemata.

Definition 3.11 *Let $\sigma \in \Sigma_m, t_1, \ldots, t_m \in F_\Sigma$ and $S_i = PSC(t_i, k, \mathbf{A}), k \geq 1$. We define the* **$\sigma$-composition** *of the schemata $S_i, i = 1, \ldots, m, \sigma(S_1, \ldots, S_m)$, as follows. First we construct a tree S (that will be the corresponding unpruned schema). The root of S is labeled by $\sigma(t_1, \ldots, t_m)$. Suppose that a node $u \in dom(S)$, $|u| \leq k-1$, is labeled by $\sigma(K_1, \ldots, K_m)$ where $K_i = S_i(v_i), i = 1, \ldots, m$, and the node $v_i \in dom(S_i)$ has r_i daughters. (Note that $t_i = S_i(\lambda), i = 1, \ldots, m$.)*

(i) Let $\sigma(K_1, \ldots, K_m)$ be existential (i.e., $K_1, \ldots, K_m$ are all existential). Then the node u has daughters labeled by all configurations $\sigma(K_1', \ldots, K_m')$ where

(ia) K_i' is a daughter of K_i (in S_i) or $K_i' = K_i$, and,

(ib) there exists at least one j such that $K_j' \neq K_j$ and $K_j' \notin A$, (i.e., K_j' is universal.)

(ic) Furthermore, if a_i is a daughter of K_i in $S_i, i = 1, \ldots, m$, and $\sigma(a_1, \ldots, a_m)$ is existential, then u has daughters labeled by all elements of $\sigma_E(a_1, \ldots, a_m)$. If $\sigma(a_1, \ldots, a_m)$ is universal, then u has a daughter labeled by $\sigma(a_1, \ldots, a_m)$. (Note

that these conditions guarantee that all daughters of $\sigma(K_1, \ldots, K_m)$ are universal configurations or elements of A.)

(ii) Suppose that $\sigma(K_1, \ldots, K_m)$ is universal and that K_j is universal iff $j \in \{i_1, \ldots, i_c\}, c \geq 1, 1 \leq i_1 < \ldots < i_c \leq m$. Then the node u has $r_{i_1} \ldots r_{i_c}$ daughters labeled by the configurations $\sigma(K'_1, \ldots, K'_m)$ where

$$K'_j = \begin{cases} K_j & \text{if } j \notin \{i_1, \ldots, i_c\} \\ \text{some daughter of } K_j & \text{if } j \in \{i_1, \ldots, i_c\}. \end{cases}$$

Furthermore if for some $\sigma(K'_1, \ldots, K'_m)$ above $K'_i = a_i \in A, i = 1, \ldots, m$, and $\sigma(a_1, \ldots, a_m)$ is universal, then the node $\sigma(K'_1, \ldots, K'_m)$ is replaced by nodes labeled by elements of $\sigma_U(a_1, \ldots, a_m)$.

Finally, if $u \in dom(S)$ and $|u| = k$, then u is a leaf of S.

Now the composition $\sigma(S_1, \ldots, S_m)$ is defined to be $pr(S)$ where pr is the pruning function defined after Definition 3.7.

Clearly the σ-composition of k-bounded schemata is a tree of height at most k such that existential and universal configurations alternate as internal nodes in each branch and all leaves are labeled by elements of A. The composition of schemata respects the composition of trees as follows.

Lemma 3.12 *Let $k \geq 1, m \geq 1, \sigma \in \Sigma_m$, and $t_1, \ldots, t_m \in F_\Sigma$. Denote $t = \sigma(t_1, \ldots, t_m), S = PSC(t, k, \mathbf{A})$ and $S_i = PSC(t_i, k, \mathbf{A}), i = 1, \ldots, m$. Then*

$$S = \sigma(S_1, \ldots, S_m).$$

Proof. This follows straightforwardly from the definition of σ-composition. In the schema S the daughters of an existential node $\sigma(K_1, \ldots, K_m)$ are all universal configurations K such that $\sigma(K_1, \ldots, K_m)(\Rightarrow_{\mathbf{A}}{}^{E})^+ K$. (Here K may be also existential if $K \in A$.) These are exactly the configurations where at least one K_i is replaced by its daughter in S_i as in Definition 3.11 (i) (where the case $K \in A$ is handled separately.)

Similarly, the daughters of a universal configuration $\sigma(K_1, \ldots, K_m)$ in S are exactly all configurations obtained from $\sigma(K_1, \ldots, K_m)$ by reading all the universal active subtrees. These are obtained from the daughters of universal configurations K_j as in Definition 3.11 (ii). (Note that if K_j is not universal then each $f \in \mathrm{act}(K_j)$ is existential and K_j necessarily remains unchanged in the universal computation segment starting from $\sigma(K_1, \ldots K_m)$.)

Finally the branches in the composition $\sigma(S_1, \ldots, S_m)$ are terminated after the k^{th} level exactly as in the schema S. Q.E.D.

Next we define the reduced simplified computation schemata that will contain all essential information about the corresponding k-bounded computation trees. Intuitively, the reduced simplified schema is obtained by removing the labels of internal nodes and then identifying identical subtrees. This means that the set of reduced simplified schemata will be finite.

Let $S = \mathrm{PSC}(t, k, \mathbf{A}), t \in F_\Sigma, k \geq 1$; the **simplified schema** corresponding to S, $\mathrm{sim}(S)$, is defined by relabeling each internal existential and universal node respectively by E and U. The **reduced simplified schema**, $\mathrm{redsim}(S)$, is obtained by identifying identical subtrees of a given node of $\mathrm{sim}(S)$ recursively in the bottom-up direction.

Set $S_0 = \mathrm{sim}(S)$. Suppose that $ui, uj \in \mathrm{dom}(S_r), r \geq 0, u \in N_+^*, i, j \in N_+, i < j$, and $S_r/ui = S_r/uj$. Furthermore we assume that $S_r/vi_1 \neq S_r/vi_2$ always when

$i_1 \neq i_2$ and u is a proper prefix of v, i.e., u is chosen to be maximal. Then one defines S_{r+1} to be the tree obtained by removing the subtree S_r/uj from S_r. There exists $C \in N_+$ such that $S_r = S_{r+1}$ always when $r \geq C$ and we define

$$\text{redsim}(S) = S_C.$$

The construction of S_{r+1} from S_r was defined nondeterministically. However because u is always chosen to be maximal, it is clear that redsim(S) is well defined. (Equivalently one could consider some fixed order for the identification process.)

Now for each $k \geq 1$, the cardinality of the set

$$D(k) = \{\text{redsim}(\text{PSC}(t, k, \mathbf{A}))|t \in F_\Sigma\}$$

is finite. Already the simplified schema sim(S), $S = \text{PSC}(t, k, \mathbf{A})$, is a tree of height at most k where the nodes are labeled by elements of $A \cup \{E, U\}$. However, the number of daughters of a given node of sim(S) is in general unbounded (since t can be arbitrary). In redsim(S) one obtains a bound for the arity of the nodes (assuming that also k is fixed). In fact, $\#D(1) \leq 2(2^{\#A} - 1) + 1 = 2^{\#A+1} - 1$ (PSC$(t, k, \mathbf{A})$ may also be the empty schema), and in general $\#D(k+1) \leq 2^{[\#D(k)+\#A+1]}$.

Thus a finite automaton can use the reduced simplified schemata to remember all possible k-bounded computation trees of the input processed so far. We still need to define the compositions of simplified schemata. This is done completely analogously with Definition 3.11. In fact, these definitions could both be obtained as special cases from a more general notion of composition of schemata. However, we presented Definition 3.11 separately because it has a very clear intuitive meaning which makes also the idea behind the next definition more transparent.

Definition 3.13 *Let $\mathbf{A} \in ATR$ and $k \geq 1$. Let $m \geq 1, \sigma \in \Sigma_m$, and $S_1, \ldots, S_m$ be simplified schemata (i.e., computation schemata where the internal nodes are labeled just by E and U). Then the composition of $S_1, \ldots, S_m$,*

$$S = \sigma(S_1, \ldots, S_m)$$

is defined by the following. First we define a tree T as follows. Nodes of T are labeled by elements of A or elements of the form $\sigma(x_1, \ldots, x_m)$ where $x_i = (S_i(u_i), u_i), u_i \in dom(S_i)$. An element $\sigma(x_1, \ldots, x_m)$ is said to be universal if

$$\text{there exists } x_i = (S_i(u_i), u_i) \text{ such that } S_i(u_i) = U, \text{ or} \tag{5}$$

$$S_1(u_1), \ldots, S_m(u_m) \in A \text{ and } \sigma(S_1(u_1), \ldots, S_m(u_m)) \text{ is a universal active subtree.} \tag{6}$$

Otherwise $\sigma(x_1, \ldots, x_m)$ is existential.

The root of T is labeled by $\sigma((S_1(\lambda), \lambda), \ldots, (S_m(\lambda), \lambda))$. Assume that a node $u \in dom(T), |u| \leq k - 1$, is labeled by an element $R = \sigma((S_1(u_1), u_1), \ldots, (S_m(u_m), u_m))$.

(i) Suppose that R is existential. Then u has daughters labeled by elements

$$\sigma((S_1(v_1), v_1), \ldots, (S_m(v_m), v_m)) \tag{7}$$

where $(a) v_i = u_i$ or $(b) v_i = u_i n, n \in N_+, u_i n \in dom(S_i)$, and for at least one $i \in \{1, \ldots, m\}$ the case (b) holds with $S_i(v_i) = U$. Furthermore, if $a_i \in A$ is a daughter of the node u_i in $S_i, i = 1, \ldots, m$, and $\sigma(a_1, \ldots, a_m)$ is an existential

active subtree of $\mathbf{A}$*, then* u *has also daughters labeled by elements of* $\sigma_E(a_1, \ldots, a_m)$*. If* $\sigma(a_1, \ldots, a_m)$ *is a universal active subtree, then* u *has a daughter labeled by* $\sigma((a_1, u_1n_1), \ldots, (a_m, u_mn_m))$ *where* $a_i = S_i(u_in_i), n_i \in N_+, i = 1, \ldots, m$.

(ii) Suppose that R *is universal. Then* u *has daughters labeled by elements*

$$\sigma((S_1(v_1), v_1), \ldots, (S_m(v_m), v_m)) \tag{8}$$

where $(a)v_i = u_i$ *if* $S_i(u_i) = E$*, and* $(b)v_i$ *is a daughter of* u_i *(in* S_i*) if* $S_i(u_i) = U, i = 1, \ldots, m$*. Furthermore if for some element as in (8),* $S_i(v_i) = a_i \in A, i = 1, \ldots, m$*, and* $\sigma(a_1, \ldots, a_m)$ *is a universal active subtree of* $\mathbf{A}$ *then this node is replaced by nodes labeled by elements of* $\sigma_U(a_1, \ldots, a_m)$.

Next we relabel the existential inner nodes of T *by* E *and the universal inner nodes by* U*. The nodes of* T *are said to be universal or existential according to (5) and (6). The labels of leaves of* T *are left unchanged in the relabeling. We denote by* T_1 *the tree obtained from* T *as the result of the relabeling.*

Now the composition S *is obtained by pruning the tree* T_1*, i.e.,*

$$\sigma(S_1, \ldots, S_m) = pr(T_1).$$

Here for the definition of the pruning function pr one considers the internal nodes of T_1 labeled by E to be existential and those labeled by U to be universal. Thus in $\sigma(S_1, \ldots, S_m)$ all leaves are labeled by elements of A and it is a simplified computation schema. Note that the composition $\sigma(S_1, \ldots, S_m)$ need not be reduced even if the schemata $S_1, \ldots, S_m$ are reduced.

Lemma 3.14 *Let* $\sigma \in \Sigma_m$*, and* $S_1, \ldots, S_m$ *be* k*-bounded computation schemata of* $\mathbf{A}, k \geq 1$*. Then*

$$\mathrm{sim}(\sigma(S_1, \ldots, S_m)) = \sigma(\mathrm{sim}(S_1), \ldots, \mathrm{sim}(S_m)).$$

Proof. This follows immediately from the Definitions 3.11 and 3.13. For the σ-composition of the computation schemata $S_1, \ldots, S_m$ (in Definition 3.11) one uses the configurations labeling the internal nodes of $S_i, i = 1, \ldots, m$, only to determine whether the node is existential or universal. Hence it does not make a difference whether the configurations are replaced by the symbols E and U before or after the composition. Q.E.D.

Lemma 3.15 *Let* $\sigma \in \Sigma_m$*, and* $S_1, \ldots, S_m$ *be simplified* k*-bounded computation schemata of* $\mathbf{A}, k \geq 1$*. Then*

$$\mathrm{red}(\sigma(\mathrm{red}(S_1), \ldots, \mathrm{red}(S_m))) = \mathrm{red}(\sigma(S_1, \ldots, S_m)).$$

Proof. Denote $R_1 = \sigma(\mathrm{red}\ (S_1), \ldots, \mathrm{red}(S_m))$ and $R_2 = \sigma(S_1, \ldots, S_m)$. Since $\mathrm{red}(S_i)$ is obtained by identifying some identical subtrees of $S_i, i = 1, \ldots, m$, it follows that R_1 is obtained from R_2 by identifying some subtrees. Thus it is clear that $\mathrm{red}(R_1) = \mathrm{red}(R_2)$. Q.E.D.

Lemma 3.16 *Let* $k \geq 1, m \geq 1, \sigma \in \Sigma_m, t_1, \ldots, t_m \in F_\Sigma$*, and denote* $t = \sigma(t_1, \ldots, t_m)$*. Then*

$$\mathrm{redsim}(\mathrm{PSC}(t, k, \mathbf{A})) = \tag{9}$$

$$\mathrm{red}(\sigma(\mathrm{redsim}(\mathrm{PSC}(t_1, k, \mathbf{A})), \ldots, \mathrm{redsim}(\mathrm{PSC}(t_m, k, \mathbf{A})))).$$

Proof. Denote $S = \mathrm{PSC}(t, k, \mathbf{A})$ and $S_i = \mathrm{PSC}(t_i, k, \mathbf{A}), i = 1, \ldots, m$. By Lemma 3.12,

$$S = \sigma(S_1, \ldots, S_m).$$

Thus by Lemma 3.14,

$$\mathrm{sim}(S) = \sigma(\mathrm{sim}(S_1), \ldots, \mathrm{sim}(S_m))$$

and (9) follows from Lemma 3.15. Q.E.D.

Now using Lemma 3.16, corresponding to an alternating recogniser $\mathbf{A}$ we can construct a deterministic tree recognizer that arrives at the root of an input tree t in the state $\mathrm{redsim}(\mathrm{PSC}(t, k, \mathbf{A}))$.

Theorem 3.17 *For every $k \geq 1$,*

$$L(\mathrm{ATR})[k] = \mathrm{REG}.$$

Proof. Clearly it is sufficient to show that $L(\mathrm{ATR})[k] \subseteq \mathrm{REG}$. Let $\mathbf{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$ and suppose that $L = L(\mathbf{A})[k], k \geq 1$.

Denote by $A\text{-SCHEMA}(k, \mathbf{A})$ the set of all k-bounded computation schemata of $\mathbf{A}, S = \mathrm{PSC}(t, k, \mathbf{A})$, such that S has an associated configuration tree with all leaves labeled by elements of A'. Now we construct a deterministic recognizer

$$\mathbf{B} = (\Sigma, B, B', H)$$

where
(i) $B = \{\mathrm{redsim}(\mathrm{PSC}(t, k+1, \mathbf{A})) | t \in F_\Sigma\}$,
(ii) $B' = \{\mathrm{redsim}(\mathrm{PSC}(t, k+1, \mathbf{A})) | t \in F_\Sigma, PSC(t, k+1, \mathbf{A}) \in A - \mathrm{SCHEMA}(k+1, \mathbf{A})\}$,
(iii) the relation H is defined by
(a) $\sigma_{E(H)} = \mathrm{redsim}(\mathrm{PSC}(\sigma, k+1, \mathbf{A}))$ if $\sigma \in \Sigma_0$,
(b) $\sigma_{E(H)}(S_1, \ldots, S_m) = \mathrm{red}(\sigma(S_1, \ldots, S_m))$,
if $m \geq 1, \sigma \in \Sigma_m, S_1, \ldots, S_m \in B$. (Here $\sigma(S_1, \ldots, S_m)$ denotes of course the σ-composition of simplified schemata.)

The set of final states B' is well defined. If $S = \mathrm{PSC}(t, k+1, \mathbf{A})$ then S is of course not determined by $\mathrm{redsim}(S)$. However, using $\mathrm{redsim}(S)$ one can determine whether $S \in A - \mathrm{SCHEMA}(k+1, \mathbf{A})$. One constructs the associated trees of $\mathrm{redsim}(S)$ by taking all successors of universal nodes and exactly one successor of an existential node. Since $\mathrm{redsim}(S)$ is obtained from S by relabeling internal nodes and identifying identical subtrees it is clear that there exists an associated tree of $\mathrm{redsim}(S)$ with all leaves labeled by elements of A' iff $S \in A - \mathrm{SCHEMA}(k+1, \mathbf{A})$. This observation also guarantees that the construction of $\mathbf{B}$ is effective.

Now we claim that for every $t \in F_\Sigma$ the recognizer $\mathbf{B}$ reaches the root of t in the state $\mathrm{redsim}(\mathrm{PSC}(t, k+1, \mathbf{A}))$. If $t \in \Sigma_0$ this follows from the definition of H. Suppose then that $m \geq 1, \sigma \in \Sigma_m, t = \sigma(t_1, \ldots, t_m)$ and the claim holds for $t_1, \ldots, t_m$. Then $\mathbf{B}$ reaches the root of t in the state

$$\begin{aligned} & \sigma_{E(H)}(\mathrm{redsim}(\mathrm{PSC}(t_1, k+1, \mathbf{A})), \ldots, \mathrm{redsim}(\mathrm{PSC}(t_m, k+1, \mathbf{A}))) \\ = \; & \mathrm{red}(\sigma(\mathrm{redsim}(\mathrm{PSC}(t_1, k+1, \mathbf{A})), \ldots, \mathrm{redsim}(\mathrm{PSC}(t_m, k+1, \mathbf{A})))) \\ = \; & \mathrm{redsim}(\mathrm{PSC}(t, k+1, \mathbf{A})). \end{aligned}$$

The second equality follows from Lemma 3.16. From Lemma 3.10 it follows that $\mathrm{redsim}(\mathrm{PSC}(t, k+1, \mathbf{A})) \in B'$ iff $t \in L(\mathbf{A})[k]$. Thus $L(\mathbf{B}) = L(\mathbf{A})[k]$. Q.E.D.

4 The logarithmic bound

In this section we show that a logarithmic alternation depth bound defines a family of forests strictly larger than the regular forests. We denote by log the function $n \to \lceil \log_2(n) \rceil$ where $\lceil \log_2(n) \rceil$ is the smallest positive integer not less than the 2-based logarithm of n.

Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{\gamma\}, \Sigma_1 = \{\sigma\}$ and $\Sigma_2 = \{\omega\}$, and denote $\Omega = \Sigma_0 \cup \Sigma_2$. Define the tree homomorphism $h : F_\Omega \to F_\Sigma$ by the following:

$$h_0(\gamma) = \gamma \text{ and } h_2(\omega) = \sigma(\omega(x_1, x_2)).$$

The Σ-tree $h(t)$ is obtained from an Ω-tree t simply by attaching above every ω-node of t a node labeled by the unary symbol σ.

Lemma 4.1 *Let Σ, Ω and h be as above and denote*

$$L = \{h(r) | r \in F_\Omega \text{ and } r \text{ is balanced}\}.$$

Then $L \in L(ATR)[2\log]$.

Proof. Clearly the set $h(F_\Omega)$ is regular. Hence by Lemma 3.4 it is sufficient to construct a recognizer $\mathbf{A} = (\Sigma, A, A', G) \in \text{ATR}$ such that

$$\text{for every } t \in h(F_\Omega) : t \in L(\mathbf{A}) \text{ if and only if } t \text{ is balanced,} \tag{10}$$

and for every balanced tree $h(r), r \in F_\Omega$,

$$\text{COM}(\mathbf{A}, h(r))[2\log] \cap \text{ ACOM}(\mathbf{A}, h(r)) \neq \emptyset. \tag{11}$$

That is, we can assume that the inputs are of the form $h(r), r \in F_\Omega$. Choose

$$\begin{aligned} A &= \{c_i, d_i, e_i, f_i, g_i | i = 1, 2, 3\}, \text{ and} \\ A' &= A - \{d_1, d_2, d_3\}. \end{aligned}$$

The state-transition relation $G = E \cup U$ is defined by the following. Below addition is always performed modulo three.

$$\gamma_E = \{c_1\}; \tag{12}$$

$$\omega_U(c_i, c_i) = \{e_i, g_i\}, i = 1, 2, 3; \tag{13}$$

$$\sigma_E(e_i) = \{c_{i+1}, d_i\}, i = 1, 2, 3; \tag{14}$$

$$\sigma_E(x) = \{x\} \text{ if } x \in \{d_1, d_2, d_3, f_1, f_2, f_3, g_1, g_2, g_3\}; \tag{15}$$

$$\omega_E(g_i, g_i) = \{g_i\}, i = 1, 2, 3; \tag{16}$$

$$\omega_E(d_i, d_i) = \{d_i\}, i = 1, 2, 3; \tag{17}$$

$$\omega_E(x, y) = \{f_i\} \text{ if } \{x, y\} = \{d_i, g_i\}, i \in \{1, 2, 3\}; \tag{18}$$

$$\omega_B(x,y) = \{f_i\} \text{ if } x,y \in \{d_i, g_i, f_i\} \text{ and } f_i \in \{x,y\}, i \in \{1,2,3\}. \quad (19)$$

The state-transition relation is undefined in all other cases. We say that a configuration $K \in F_\Sigma(A)$ is well formed if $K = h(r)$ for some ΩA-tree r. (The tree homomorphism h is extended to $F_\Omega(A)$ by setting $h(a) = a$ for all $a \in A$.) Let $K \in F_\Sigma(\{a_1, \ldots, a_n\})$ be such that K does not contain the nullary symbol γ, and $a_1, \ldots, a_n \in A$. If each element $a_i, i = 1, \ldots, n$, occurs at least once in K, it is called an $[a_1, \ldots, a_n]$-configuration. First we show that (11) holds.

Claim 1 *Let K_1 be a well formed balanced $[c_i]$-configuration, $i \in \{1,2,3\}$, and denote $m = \mathrm{hg}(K_1)$. We claim that there exists $T \in \mathrm{ACOM}(\mathbb{A}, K_1)$ such that $\mathrm{alt}(T) \leq m$.*

Proof of Claim 1. Since K_1 is well formed, each subtree of K_1 of height two is of the form $\sigma(\omega(c_i, c_i))$. In the computation of T the recogniser reads first all (universal) active subtrees $\omega(c_i, c_i)$ in arbitrary order using the rule (13). Thus one obtains one $[e_i]$-configuration $K(1)$, one $[g_i]$-configuration $K(2)$ and a number of $[e_i, g_i]$-configurations $K(3)$. In each configuration $K(3)$ the recogniser reads all active subtrees $\sigma(e_i)$ making the existential choice d_i, this results in a $[d_i, g_i]$-configuration $K(4)$. Since $K(4)$ contains both states d_i and g_i, it follows that the recogniser reaches the root of $K(4)$ in the accepting state f_i by the deterministic rules (15)-(19). Similarly the computation starting from $K(2)$ reaches the root in the state g_i using rules (15) and (16). Finally, in $K(1)$ the recogniser makes in each active subtree $\sigma(e_i)$ the existential choise c_{i+1}, which yields a $[c_{i+1}]$-configuration K_2. Furthermore K_2 is balanced because K_1 is balanced.

Above the computations starting from configurations $K(2)$, $K(3)$ and $K(4)$ are purely existential. Thus there exists a K_1-computation tree T_1 such that $\mathrm{alt}(T_1) = 2$ and one leaf of T_1 is labeled by K_2 and all other leaves by elements of A'. Now $\mathrm{hg}(K_2) = \mathrm{hg}(K_1) - 2$. By inductive reasoning it follows that T_1 can be completed to an accepting computation tree T, where $\mathrm{alt}(T) = m = \mathrm{hg}(K_1)$. (Since K_1 is well formed, m is even. The configuration $K_{(m/2+1)}$ will be of the form $c_j, j \in \{1,2,3\}$.) This concludes the proof of the claim.

Now let $t \in h(F_\Omega)$ be balanced. We construct $T \in \mathrm{COM}(\mathbb{A}, t)$ as follows. First the recogniser reads the leaves of t using the rule (11) yielding a $[c_1]$-configuration K_1. By Claim 1 there exists an accepting K_1-computation tree T_1 such that $\mathrm{alt}(T_1) = \mathrm{hg}(K_1)(= \mathrm{hg}(t))$ where furthermore in each branch the first computation segment is universal. Thus T can be constructed so that

$$\mathrm{alt}(T) = \mathrm{hg}(t) + 1.$$

(The first computation segment corresponding to rules (12) is existential.) Since t is balanced, $\mathrm{hg}(t) < 2\log(\mathrm{size}(t))$ and (12) holds.

It remains to verify that also (10) holds. The "if" direction follows from (11). The intuitive idea of the proof in the "only if" direction is to show that in an accepting t-computation tree there necessarily exists a branch where the recogniser essentially reads the input in a layered fashion as in the proof of Claim 1 and thus checks that the input tree t is balanced. First we prove a number of claims. Denote

$$Q_i = \{c_i, d_i, g_i, f_i\}, i = 1,2,3.$$

Claim 2 *Let $K \in F_\Sigma(A)$ and assume that K contains elements of Q_i and Q_j, $i \neq j$. Then K is not accepting.*

Proof of Claim 2. Let $T \in \mathrm{COM}(\mathbf{A}, K)$ be arbitrary and let H label a node of T. If H contains an element of $Q_k, k \in \{1,2,3\}$, then one daughter of H in T also contains an element of Q_k. This follows immediately from the definition of the rules that read elements of Q_k, note that in rule (13) one can choose the daughter corresponding to g_k. Now since K contains elements of both Q_i and Q_j, the computation tree T contains a branch where each configuration has elements of Q_i and Q_j and this computation cannot terminate successfully.

Claim 3 *Let $K \in F_\Sigma(A)$ and assume that states e_i and c_{i+2} appear in K, ($i+2$ is computed modulo 3.) Then K is not accepting.*

Proof of Claim 3. We can assume that e_i appears in an active subtree $r_1 = \sigma(e_i)$ and c_{i+2} in an active subtree $r_2 = \omega(c_{i+2}, c_{i+2})$ because otherwise the computation is blocked already in the states in question. Let T be an arbitrary K-computation tree of $\mathbf{A}$. Assume that in T the recogniser reads r_1 before r_2. The existential rule (14) yields the state c_{i+1} or d_i which cannot appear together with c_{i+2} in an accepting configuration by Claim 2. Thus necessarily the recogniser reads first r_2 using the universal rule (13). Consider the branch of the computation corresponding to the state g_{i+2}. From rules (15), (16), (18) and (19) it follows that all configurations in this branch contain one of the states g_{i+2} or f_{i+2}, (g_{i+2} can only be deleted by changing it to f_{i+2} using rule (18) or (19).) So when the recogniser reads the active subtree r_1 at an arbitrary time in the computation, both existential choices c_{i+1} and d_i yield a configuration that is not accepting by Claim 2. Thus $T \notin \mathrm{ACOM}(\mathbf{A}, K)$.

Claim 4 *Denote $D = \{c_1, c_2, c_3, e_1, e_2, e_3, d_1, d_2, d_3\}$. Assume that all leaves of a configuration K are labeled by elements of D and K contains at least one element of $\{d_1, d_2, d_3\}$. Then K is not accepting.*

Proof of Claim 4. Let $T \in \mathrm{COM}(\mathbf{A}, K)$ be arbitrary. Consider the branch B of T that in universal computation steps (13) follows the choice e_i. All configurations in this branch have leaves labeled by elements of D and furthermore contain at least one element of $\{d_1, d_2, d_3\}$. This is because the elements d_i can be deleted only by rules (18) and (19) which do not become applicable as the configurations do not contain elements g_i or f_i. Thus B cannot end with an accepting final state.

Claim 5 *Let K be an $\mathbf{A}$-configuration with all leaves labeled by $e_i, i \in \{1,2,3\}$, and $hg(K) > 1$. Assume that T is an accepting K-computation tree of $\mathbf{A}$. Then T contains a configuration K_1 with all leaves labeled by e_{i+1}. Furthermore,*

$$K = K_1(e_{i+1} \leftarrow \omega(\sigma(e_i), \sigma(e_i))). \tag{20}$$

Proof of Claim 5. Necessarily the mother (immediate predecessor) of each node e_i is labeled by σ because otherwise the computation would be blocked in the state e_i. The computation of T first reads an arbitrary number of the active subtrees $\sigma(e_i)$ making the choice c_{i+1}. The existential choice d_i is prohibited by Claim 4. In the states c_{i+1} the recogniser can then apply only the universal rule (13). Consider the branch B of the computation that corresponds to universal choices e_{i+1}. Note that above the recogniser needs not read all subtrees $\sigma(e_i)$ before starting to read the subtrees $\omega(c_{i+1}, c_{i+1})$. However, before continuing the computation from the states e_{i+1} the recogniser must read all active subtrees $\sigma(e_i)$ and $\omega(c_{i+1}, c_{i+1})$. This is seen as follows.

The state e_{i+1} can only be read by rule (14) where by Claim 4 furthermore the recogniser needs to make the existential choice c_{i+2}. (The current configuration

contains only states e_i, c_{i+1}, e_{i+1}.) By Claims 2 and 3, the state c_{i+2} cannot appear with c_{i+1} or e_i in an accepting configuration. Thus we can choose K_1 to be the configuration that appears in the branch B just before the first symbol e_{i+1} is read. Then all leaves of K_1 are labeled by e_{i+1} and also clearly (20) holds. The assumption $\mathrm{hg}(K) > 1$ prohibits the possibility that $K = \sigma(e_i)$.

Now we can proceed to prove that the "only if" side of (10) holds. Assume that $t \in L(\mathrm{A})$ and let $T \in \mathrm{COM}(\mathrm{A}, t)$ be accepting. Without restriction we can assume that in T the recognizer first reads all leaf symbols γ. Note that the rule (12) is deterministic so it commutes with all other rules. Thus one obtains a configuration $K = t(\gamma \leftarrow c_1)$. Consider the branch of T that corresponds to the universal choices e_1 in the computation steps (13) in the configuration K. (Note that (13) is the only computation step applicable in K.) In this branch of the computation the recognizer must read all states c_1 before reading any of the states e_1 by rule (14). (This is seen using Claims 2-4 exactly as in the proof of Claim 5.) Thus one obtains a configuration K_1 with all leaves labeled by e_1 such that

$$t = K_1(e_1 \leftarrow \omega(\gamma, \gamma)).$$

Denote by $\lfloor j \rfloor$ the smallest positive integer congruent to j modulo 3. By Claim 5, if $K_i \in \mathrm{conf}(T)$ is a configuration with all leaves labeled by $e_{\lfloor i \rfloor}$ and having height at least two, there exists a configuration $K_{i+1} \in \mathrm{conf}(T)$ with all leaves labeled by $e_{\lfloor i+1 \rfloor}$ such that

$$K_i = K_{i+1}(e_{\lfloor i+1 \rfloor} \leftarrow \omega(\sigma(e_{\lfloor i \rfloor}), \sigma(e_{\lfloor i \rfloor}))). \tag{21}$$

Denote $m = (\mathrm{hg}(K_1) + 1)/2$. Then $K_m = \sigma(e_{\lfloor m \rfloor})$. (Since $t \in h(F_\Omega)$, it is easy to see that the string of symbols labeling a path from a leaf of K_1 to the root always belongs to $(\sigma\omega)^*\sigma$. Hence $\mathrm{hg}(K_1)$ is odd and the last configuration in the chain defined by (21) is $\sigma(e_{\lfloor m \rfloor})$.) From (21) it follows that K_1 and hence t is balanced. Q.E.D.

In Lemma 4.1 the function 2 log can be reduced by an arbitrary constant factor. The construction in the proof is independent of the rank of the elements ω and by increasing $\mathrm{rank}(\omega)$ the number of distinct existential and universal segments in a computation on an input t can be made to be smaller than $C^{-1}\log(\mathrm{size}(t))$ for any natural number C.

Let $m \geq 2$ and define $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_m$, where $\Gamma_0 = \{\gamma\}, \Gamma_1 = \{\sigma\}$ and $\Gamma_m = \{\overline{\omega}\}$, i.e., Γ is as Σ in Lemma 4.1 except the binary symbol ω is replaced by $\overline{\omega}$ of rank m. Define $L(m)$ to be the Γ-forest that is obtained from the forest L of Lemma 4.1 by relabeling each ω-node with $\overline{\omega}$ and attaching for it $m-2$ additional copies of the subtrees. In other words, $L(m)$ consists of all balanced Γ-trees t such that the string of labels of each branch from the root of t to a leaf belongs to $(\sigma\overline{\omega})^*\gamma$. Define $\mathrm{A} = (\Gamma, A, A', G(m)) \in \mathrm{ATR}$ otherwise exactly as in Lemma 4.1 except the rules (13), (16), (17), (18) and (19) are replaced by the following:

$$\overline{\omega}_U(c_i, \ldots, c_i) = \{e_i, g_i\}, i = 1, 2, 3; \tag{13}'$$

$$\overline{\omega}_E(g_i, \ldots, g_i) = \{g_i\}, i = 1, 2, 3; \tag{16}'$$

$$\overline{\omega}_E(d_i, \ldots, d_i) = \{d_i\}, i = 1, 2, 3; \tag{17}'$$

$$\overline{\omega}_E(x_1, \ldots, x_m) = \{f_i\} \text{ if } \{x_1, \ldots, x_m\} = \{d_i, g_i\}, i \in \{1, 2, 3\}; \tag{18}'$$

$$\overline{\omega}_E(x_1, \ldots, x_m) = \{f_i\} \text{ if } x_1, \ldots, x_m \in \{d_i, g_i, f_i\}, \text{ and} \tag{19}'$$

$$f_i \in \{x_1, \ldots, x_m\}, i \in \{1, 2, 3\}.$$

Then exactly as in the proof of Lemma 4.1 it is seen that

$$L(\mathbb{A}) = L(m),$$

and furthermore for every $t \in L(m)$ there exists $T \in \mathrm{ACOM}(\mathbb{A}, t)$ such that $\mathrm{alt}(T) = \mathrm{hg}(t) + 1$. Let $C \in N_+$ be arbitrary and choose $m > 2^C$. Then for every $t \in L(m)$,

$$\mathrm{hg}(t) < (2/C)\log(\mathrm{size}(t)).$$

Denote by $C^{-1}\log$ the function $n \to \lceil C^{-1}\log_2(n)\rceil$. Thus we have:

Theorem 4.2 *For every $C \in N_+$:*

$$REG \subset L(ATR)[C^{-1}\log].$$

(Here $\subset$ denotes strict inclusion.)

5 Conclusions

Here we briefly discuss open questions and results on other types of alternation bounds. We have shown that

$$\mathrm{REG} = L(\mathrm{ATR})[k] \subset L(\mathrm{ATR})[C^{-1}\log]$$

for all constants k and C. A central open question is whether it is possible to separate $L(\mathrm{ATR})[\theta]$ from REG for some sublogarithmic function θ. Also we do not know whether $L(\mathrm{ATR})[\log] \subset L(\mathrm{ATR})$. We conjecture that the simple forest $L = \{\omega(\sigma^n(\gamma), \sigma^n(\gamma)) | n > 0\}$ does not belong to $L(\mathrm{ATR})[\log]$ but do not have a proof for this. It is easy to see that $L \in L(\mathrm{ATR})$, cf. [14], [15].

One can restrict the computation trees of an alternating recognizer in many different ways. A natural variant of Definition 3.3 would be to require that the number of distinct existential and universal computation segments corresponding to any given path from a leaf to the root in the input tree is bounded by some function θ. Similarly as in Definitions 3.2 and 3.3, in every branch of a computation tree one can associate a word w over $\{E, U\}$ to the computation steps performed on a given path from a leaf to the root in the input tree t. Then one requires that for all such words w, $\mathrm{alt}(w)$ is at most $\theta(\mathrm{size}(t))$. With this definition it is not difficult to see that already a constant bound (in fact even the constant 2) allows the alternating automata to recognize forests that are not regular. The detailed construction is omitted here. Note that since the computations in independent subtrees can be performed in arbitrary order, a t-computation tree may have $O(\mathrm{size}(t))$ computation segments (in the sense of Definition 3.2) even if the computation on any fixed path of t has only 2 segments.

Also one can restrict the width of the computation trees or, equivalently, the number of universal computation steps analogously with the bounds on parallelism considered in [5], [7]. Let θ be a function on the natural numbers, $\mathbb{A} = (\Sigma, A, A', G) \in \mathrm{ATR}$ and T be a computation tree of $\mathbb{A}$. We denote by $\#T$ the number of leaves of T. We say that the recogniser $\mathbb{A}$ accepts a Σ-forest L with the width-bound θ if $L = L(\mathbb{A})$ and for every $t \in L$ there exists $T \in \mathrm{ACOM}(\mathbb{A}, t)$ such that $\#T \leq \theta(\mathrm{size}(t))$. This is denoted $L = L(\mathbb{A})[\theta]_w$. The family of forests recognized with the width bound θ is denoted $L(\mathrm{ATR})[\theta]_w$. As a corollary of Theorem 3. 17 we have:

Theorem 5.1 $L(\mathrm{ATR})[k]_w = \mathrm{REG}$. *(Again we denote the constant function $c(k)$ simply by k.)*

Proof. Suppose that $L = L(\mathbf{A})[k]_w$, $\mathbf{A} = (\Sigma, A, A', G)$. Without restriction we can assume that if f is an active subtree of type Z of $\mathbf{A}$, $Z \in \{E, U\}$, and f_Z consists of only one element of A, then f is existential, i.e., $Z = E$. (A suitable modification of the relation G does not change the number of leaves of any computation tree.) Thus for every computation tree T of $\mathbf{A}$ we have

$$\mathrm{alt}(T) \leq 2(\#T) - 1.$$

($\#T$ is at least the number of universal computation steps in T plus one.) Thus $L = L(\mathbf{A})[2k-1]$ and L is regular by Theorem 3.17. Q.E.D.

Also the question whether $L(\mathrm{ATR})[\log]_w$ contains nonregular forests remains open. Note that this does not follows from the results of the previous section because in the construction of Lemma 4.1 the recognizer uses $O(\mathrm{size}(t))$ universal computation steps on an input t.

References

[1] G. Buntrock and A. Hoene, Reversals and alternation, Proc. of 6th STACS, Lect. Notes Comput. Sci. 349 (1989) 218-228.

[2] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, J. Assoc. Comput. Mach. 28 (1981) 114-133.

[3] E.M. Gurari and O.H. Ibarra, (Semi)Alternating stack automata, Math. Systems Theory 15 (1982) 211-224.

[4] F. Gécseg and M. Steinby, Tree automata, Akadémiai Kiadó, Budapest, 1984.

[5] J. Hromkovic, Tradeoffs for language recognition on parallel computing models, Proc. of 13th ICALP, Lect. Notes Comput. Sci. 226 (1986) 157-166.

[6] J. Hromkovic, On the power of alternation in automata theory, J. Comput. System Sci. 31 (1985) 28-39.

[7] K.N. King, Measures of parallelism in alternating computation trees, Proc. of 13th Ann. ACM Symp. on Theory of Computing (1981) 189-201.

[8] K.N. King, Alternating multihead finite automata, Theoret. Comput. Sci. 61 (1988) 149-174.

[9] R.E. Ladner, R.J. Lipton and L.J. Stockmeyer, Alternating pushdown and stack automata, SIAM J. Comput. 13 (1984) 135-155.

[10] H. Matsuno, K. Inoue, H. Taniguchi and I. Takanami, Alternating simple multihead finite automata, Theoret. Comput. Sci. 36 (1985) 291-308.

[11] D.E. Muller, A. Saoudi and P.E. Schupp, Alternating automata, the weak monadic theory of the tree, and its complexity, Proc. of 13th ICALP, Lect. Notes Comput. Sci. 226 (1986) 275-283.

[12] D.E. Muller and P.E. Schupp, Alternating automata on infinite trees, Theoret. Comput. Sci. 54 (1987) 267-276.

[13] W.L. Ruzzo, Tree-size bounded alternation, J. Comput. System Sci. 21 (1980) 218-235.

[14] K. Salomaa, Alternating bottom-up tree recognizers, Proc. of 11th CAAP, Lect. Notes Comput. Sci. 214 (1986) 158-171.

[15] K. Salomaa, Yield-languages recognized by alternating tree recognizers, RAIRO Inform. Théor. 22 (1988) 319-339.

[16] K. Salomaa, Alternating tree pushdown automata, Ann. Univ. Turku Ser. AI 192 (1988).

[17] K. Salomaa, Representation of recursively enumerable languages using alternating finite tree recognizers, Proc. of 7th FCT, Lect. Notes Comput. Sci. 380 (1989) 372-383.

[18] G. Slutzki, Alternating tree automata, Proc. of 8th CAAP, Lect. Notes Comput. Sci. 159 (1983) 392-404.

Received June 20, 1991.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

Initial and Final Congruences

Virgil Emil Căzănescu Rodica Ceterchi*

Abstract

The paper contains some results which establish connections between different types of algebraic structures which appeared in the process of algebrization of the theory of flowchart schemes [3,4,5,6,7]. Two basic constructions [2,4] are used, first separately, and afterwards combinations between them or their duals, in order to obtain the theorems.

1 Introduction

We recall the definitions of the algebraic structures which will be used subsequently in the paper.

Let B be a category whose objects form a monoid $(M, +, 0)$ and such that for each $a, b, c, d \in M$ a sum operation is given

$$+ : B(a, b) \times B(c, d) \longrightarrow B(a + c, b + d)$$

B is called a *strict monoidal category* (smc for short) if Axioms 1-4 from Table 1 are satisfied. If B satisfies the weaker axioms 1, 2, 3, 4a and 4b of Table 1, then B is called a *nonpermutable strict monoidal category* (nsmc for short).

$1.f + (g + h) = (f + g) + h$
$2.f + I_0 = f = I_0 + f$
$3.I_a + I_b = I_{a+b}$

$4.(f + g)(u + v) = fu + gv$

$4a.(f + g)(I_b + v) = f + gv$
$4b.(f + I_d)(u + v) = fu + v$
$4c.(I_{c+d} + f)({}^cX^d + I_b) = {}^cX^d + f$

$5.{}^aX^c(g + f){}^dX^b = f + g$
for $f : a \longrightarrow b$ and $g : c \longrightarrow d$

$5a.{}^aX^c(I_c + f){}^cX^b = f + I_c$
for $f : a \longrightarrow b$

$6.{}^aX^0 = I_a$
$7.{}^aX^{b+c} = ({}^aX^b + I_c)(I_b + {}^aX^c).$

Table 1: Axioms for ssmc and snsmc

*University of Bucharest, Faculty of Mathematics 14, Academiei str. 70109 Bucharest, Romania

Suppose that for every $a, b \in M$ some distinguished morphisms ${}^aX^b \in B(a+b, b+a)$ are given. An smc is called *symmetric* (ssmc for short) if Axioms 5, 6, 7 of Table 1 are satisfied. An nsmc is called *symmetric* (*snsmc* for short) if Axioms 4c, 5a, 6 and 7 are satisfied. Obviously, every ssmc is an snsmc.

Let us notice that an snsmc which satisfies *either* Axiom 4 *or* Axiom 5 of Table 1 is an ssmc. Moreover, this remains true if we replace Axiom 4 (or Axiom 5) with the following weaker axiom:

$$f + g = (I_a + g)(f + I_d)$$

for every $f \in B(a, b)$ and $g \in B(c, d)$.

In an snsmc an *$a\alpha$-morphism* is a composite of morphisms of the form $I_a + {}^bX^c + I_d$. Axiom 4 of Table 1 is satisfied in an snsmc for g or u $a\alpha$-morphisms. In an snsmc B we denote by B_a the subcategory of $a\alpha$-morphisms and we notice that B_a is an ssmc. Note also that every $a\alpha$-morphism is an isomorphism, its inverse being also an $a\alpha$-morphism.

The concept of an xy-ssmc depends on two parameters, $x \in \{a, b, c, d\}$ and $y \in \{\alpha, \beta, \gamma, \delta\}$. For every $e \in M$ we will use the distinguished morphisms $\top_e \in B(0, e)$, $\bot^e \in B(e, 0), \vee_e \in B(e+e, e)$ and $\wedge^e \in B(e, e+e)$. Table 2 shows for every value of the parameters the distinguished morphisms which are involved. The axioms which are satisfied by the distinguished morphisms are chosen from Table 3 for each xy case according to the rule: select all those axioms (and only those) in which the distinguished morphisms of the xy case appear.

x	distinguished morphisms	y	distinguished morphisms
a	none	α	none
b	$\bot^e$	β	$\top_e$
c	$\wedge^e$	γ	$\vee_e$
d	$\bot^e$ and $\wedge^e$	δ	$\top_e$ and $\vee_e$

Table 2: Distinguished morphisms for xy-ssmc and xy-snsmc

Thus, an ssmc (snsmc) will be called an *xy-ssmc (xy-snsmc)* if every object $e \in M$ is endowed with the distinguished morphisms which appear in Table 2 corresponding to the xy case and if all axioms of Table 3 which contain only these distinguished morphisms are fulfilled.

Notice that in an xy-ssmc axioms $\mathbf{P}\top, \mathbf{P}\bot, \mathbf{P}\vee$, and $\mathbf{P}\wedge$ are automatically satisfied as a consequence of Axiom 4 in Table 1.

$A.(\vee_a + I_a)\vee_a = (I_a + \vee_a)\vee_a$
$B.{}^aX^a\vee_a = \vee_a$
$C.(\top_a + I_a)\vee_a = I_a$
$D.\vee_a \bot^a = \bot^a + \bot^a$

$A^\circ.\wedge^a(\wedge^a + I_a) = \wedge^a(I_a + \wedge^a)$
$B^\circ.\wedge^a {}^aX^a = \wedge^a$
$C^\circ.\wedge^a(\bot^a + I_a) = I_a$
$D^\circ.\top_a\wedge^a = \top_a + \top_a$

$E.\top_a\bot^a = I_0$
$F.\vee_a\wedge^a = (\wedge^a + \wedge^a)(I_a + {}^aX^a + I_a)(\vee_a + \vee_a)$
$G.\wedge^a\vee_a = I_a$

$SV1.\top_0 = I_0$
$SV2.\top_{a+b} = \top_a + \top_b$
$SV3.\vee_0 = I_0$
$SV4.\vee_{a+b} = (I_a + {}^bX^a + I_b)(\vee_a + \vee_b)$

$SV1^\circ.\bot^0 = I_0$
$SV2^\circ.\bot^{a+b} = \bot^a + \bot^b$
$SV3^\circ.\wedge^0 = I_0$
$SV4^\circ.\wedge^{a+b} = (\wedge^a + \wedge^b)(I_a + {}^aX^b + I_b)$

$P\top.g(\top_a + I_c) = \top_a + g$
$P\vee.(I_{a+a} + g)(\vee_a + I_c) = \vee_a + g$
for $g : b \longrightarrow c$

$P\bot.(I_a + g)(\bot^a + I_c) = \bot^a + g$
$P\wedge.(I_a + g)(\wedge^a + I_c) = \wedge^a + g$

Table 3: Axioms for xy-ssmc and xy-snsmc

$S\top.\top_a f = \top_b$
$S\vee.(f + f)\vee_b = \vee_a f$

$S\bot.f\bot^b = \bot^a$
$S\wedge.\wedge^a(f + f) = f\wedge^b$

for $f : a \longrightarrow b$

Table 4: Axioms for strong xy-ssmc

Let us consider the order $<_L$ on $\{a, b, c, d\}$ given by $a <_L b <_L d$ and $a <_L c <_L d$, and the same order $<_G$ on the corresponding Greek letters, so that, e.g. $\alpha <_G \beta <_G \delta$. For $x' \leq_L x$ and $y' \leq_G y$, we define an *$x'y'$-strong xy-ssmc* to be an xy-ssmc in which all the axioms in Table 4 corresponding to the $x'y'$ case hold. A *strong xy-ssmc* will be, by definition, an xy-strong xy-ssmc.

Suppose that in an snsmc B we are given, for each $a, b, c \in M$, an unary operation

$$\uparrow^a\_ : B(a + b, a + c) \longrightarrow B(b, c)$$

called *(left) feedback*.

A *biflow (flow)* is an ssmc (snsmc) endowed with a feedback which satisfies all the axioms in Table 5. As we will use sometimes the right feedback $\_\uparrow^a$: $B(b + a, c + a) \longrightarrow B(b, c)$ we mention the connections between the two feedbacks:

$$f\uparrow^a = \uparrow^a({}^aX^b f\, {}^aX^a) \qquad \text{for } f : b + a \longrightarrow c + a.$$

$$\uparrow^a f = ({}^bX^a f\, {}^aX^c)\uparrow^a \quad \text{for } f : a + b \longrightarrow a + c.$$

1. $f(\uparrow^a g)h = \uparrow^a [(I_a + f)g(I_a + h)]$
2. $\uparrow^a f + I_d = \uparrow^a (f + I_d)$
3. $\uparrow^{a+b} [f({}^bX^a + I_d)] = \uparrow^{b+a} [({}^bX^a + I_c)f]$
   for $f : a + b + c \longrightarrow b + a + d$
4. $\uparrow^a \uparrow^b f = \uparrow^{b+a} f$
5. $\uparrow^a I_a = I_0$
6. $\uparrow^a {}^aX^a = I_a$.

Table 5: Axioms for feedback

All the algebraic structures previously defined form categories whose morphisms are functors which are monoid morphisms on objects and which preserve the additional algebraic structure. Sometimes we will be interested in certain subcategories, namely those in which the monoid of objects, M, is kept fixed (called M-smc, M-nsmc,..., M-biflow), and where the morphisms are object-preserving functors (called M-smc morphism, ..., M-biflow morphisms). These subcategories are equational varieties in the sense of the many-sorted universal algebra. Examples of the above algebraic structures may be found in [4,5,6].

Some of the results in the rest of the paper provide the construction of left adjoints for the following forgetful functors:

a) for $x \in \{b, d\}$ and $y \in \{\alpha, \beta, \gamma, \delta\}$
$b\alpha$-strong xy-ssmc $\longrightarrow$ xy-ssmc
biflow over a $b\alpha$-strong xy-ssmc $\longrightarrow$ biflow over an xy-ssmc

b) for $x \in \{b, d\}$ and $y \in \{\beta, \delta\}$
$b\beta$-strong xy-ssmc $\longrightarrow$ xy-ssmc
biflow over a $b\beta$-strong xy-ssmc $\longrightarrow$ biflow over an xy-ssmc

c) $b\alpha$-ssmc $\longrightarrow$ ssmc
biflow over a $b\alpha$-ssmc $\longrightarrow$ biflow

d) for $y \in \{\alpha, \beta, \gamma, \delta\}$
$b\alpha$-strong by-ssmc $\longrightarrow$ ay-ssmc
strong by-ssmc $\longrightarrow$ strong ay-ssmc
biflow over a $b\alpha$-strong by-ssmc $\longrightarrow$ biflow over an ay-ssmc
biflow over a strong by-ssmc $\longrightarrow$ biflow over a strong ay-ssmc

e) $b\alpha$-ssmc and $a\beta$-ssmc $\longrightarrow$ ssmc
biflow over a $b\alpha$-ssmc and $a\beta$-ssmc $\longrightarrow$ biflow.

Even if the existence of left adjoints follows from general principles, our constructions are more effective than the general ones.

Table 6 represents the structure of the paper as a graph, where each section (vertex) depends only on the sections at higher levels with which it is linked.

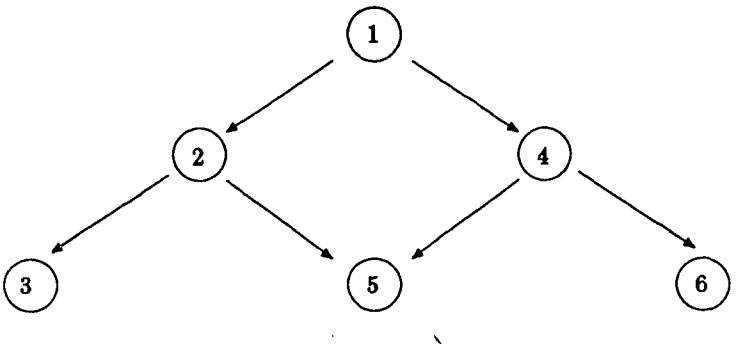

Table 6: The structure of the paper

2 Final congruences

In this section we will show that the congruence relation introduced by Bloom and Tindell [2] in an algebraic theory is useful also in the study of more general algebraic structures. We mention that, instead of the name "zero-congruence" used in [2], we will use the name "final congruence", one reason being that it can be dualized into "initial congruence".

In an smc B a congruence relation $\equiv$ is called *final* if $f \equiv g$ for every $f, g \in B(a,0)$. Notice that if 0 is a weak final object in B, i.e. $B(a,0) \neq \emptyset$ for every $a \in M$, then factorization with a final congruence makes 0 a final object.

Definition 2.1 [2] *For every $f, g \in B(a,b)$ we have $f\mathbf{P}g$ iff there exist: an object $x \in M$ and morphisms $h \in B(a, b+x)$ and $u, v \in B(x,0)$, such that*

$$f = h(I_b + u) \text{ and } g = h(I_b + v).$$

Since a final congruence relation identifies any two morphisms u and v in $B(x,0)$ it obviously includes the relation $\mathbf{P}$.

Note that for every $a, b \in M$ the relation $\mathbf{P}$ is reflexive and symmetric on $B(a,b)$.

The above defined relation was introduced in [2], and was used for different purposes in [1].

Lemma 2.2 *In an ssmc, relation $\mathbf{P}$ is compatible with composition and sum.*

Proof. With the notations of Definition 2.1, let $f\mathbf{P}g$.

a) Compatibility with composition. Suppose $p\mathbf{P}q$ in $B(b,c)$, that is there exist $j \in B(b, c+y)$ and $w, t \in B(y,0)$ such that $p = j(I_c + w)$ and $q = j(I_c + t)$. Notice that

$$fp = h(I_b + u)j(I_c + w) = [h(j + I_x)](I_c + w + u)$$

and similarly,

$$gq = [h(j + I_x)](I_c + t + v).$$

b) Compatibility with sum. Suppose $p\mathbf{P}q$ in $B(c,d)$, i.e. there exist $j \in B(c, d+y)$ and $w,t \in B(y,0)$ such that $p = j(I_d + w)$ and $q = j(I_d + t)$. Notice that

$$f + p = (h + j)(I_b + u + I_d + w) = [(h + j)(I_b + {}^xX^d + I_y)](I_{b+d} + u + w)$$

and similarly,

$$g + q = [(h + j)(I_b + {}^xX^d + I_y)](I_{b+d} + v + t).$$

It is known that in a Σ-algebra the transitive closure of a reflexive and symmetric relation, compatible with the operations, is a congruence. This implies that $\mathbf{P}^+$, the transitive closure of $\mathbf{P}$, is an ssmc congruence.

Proposition 2.3 *The congruence $\mathbf{P}^+$ is the least final congruence.*

To simplify the notation in the following, each time $\mathbf{R}$ is a reflexive and symmetric relation on A, we will denote by $A/\mathbf{R}$ the quotient of A by $\mathbf{R}^+$.

Proposition 2.4 *If B is a $b\alpha$-ssmc, then $B/\mathbf{P}$ is a strong $b\alpha$-ssmc. If C is a strong $b\alpha$-ssmc and $G : B \longrightarrow C$ is a $b\alpha$-ssmc morphism, then there exists a unique $b\alpha$-ssmc morphism $H : B/\mathbf{P} \longrightarrow C$ such that $G = F_B \bullet H$, where $F_B : B \longrightarrow B/\mathbf{P}$ is the canonical factorization morphism.*

Proof. It is sufficient to notice that $f\mathbf{P}g$ implies $G(f) = G(g)$.

The above proposition tells us that the forgetful functor from strong $b\alpha$-ssmc to $b\alpha$-ssmc has a left adjoint. The same construction can be used in other cases as well, giving us the left adjoints for the forgetful functors from the categories of $b\alpha$-strong xy-ssmc to the categories of xy-ssmc, for every $x \in \{b,d\}$ and every $y \in \{\alpha,\beta,\gamma,\delta\}$. If we note furthermore that in a biflow, relation $\mathbf{P}$ is compatible with the feedback, then the above result remains true for biflows over an xy-ssmc.

3 Congruences which are simultaneously initial and final

The concept of initial congruence is dual to that of final congruence, and the construction and results of the previous section can be readily dualized.

The purpose of this section is to show that the factorization from the previous section and its dual can be merged into a single factorization.

Definition 3.1 *In an ssmc B we define relation $\mathbf{R}$ by the following: $f\mathbf{R}g$ in $B(a,b)$ iff there exist objects $x,y \in M$ and morphism $h \in B(a+y, b+x), u,v \in B(x,0)$ and $p,q \in B(0,y)$ such that we have decompositions*

$$\begin{aligned} f &= (I_a + p)h(I_b + u) \\ g &= (I_a + q)h(I_b + v). \end{aligned}$$

Obviously, $\mathbf{R}$ is reflexive and symmetric. Since an initial and final congruence relation identifies any two morphisms u and v in $B(x,0)$ and any two morphisms p and q in $B(0,y)$ it obiously includes the relation $\mathbf{R}$.

Lemma 3.2 *The relation $\mathbf{R}$ is compatible with composition and sum.*

Proof. With the above notation let $f\mathbf{R}g$ in $B(a,b)$.

a) Compatibility with composition. Suppose we have $f'\mathbf{R}g'$ in $B(b,c)$, i.e. there exist $h' \in B(b+y', c+x')$, $u', v' \in B(x', 0)$, $p', q' \in B(0, y')$ such that $f' = (I_b + p')h'(I_c + u')$ and $g' = (I_b + q')h'(I_c + v')$.

It follows that

$$\begin{aligned} ff' &= (I_a + p)h(I_b + u)(I_b + p')h'(I_c + u') \\ &= (I_a + p + p')(h + I_{y'})(I_b + u + I_{y'})h'(I_c + u') \\ &= [I_a + (p + p')][(h + I_{y'})(I_b + {}^{x}X^{y'})(h' + I_x)][I_c + (u' + u)], \end{aligned}$$

and similarly,

$$gg' = [I_a + (q + q')][(h + I_{y'})(I_b + {}^{x}X^{y'})(h' + I_x)][I_c + (v' + v)],$$

which imply $ff'\mathbf{R}gg'$.

b) Compatibility with sum. Suppose that $f'\mathbf{R}g'$ in $B(c,d)$, i.e. there exist $h' \in B(c + y', d + x'), u', v' \in B(x', 0), p', q' \in B(0, y')$ such that

$$f' = (I_c + p')h'(I_d + u') \text{ and } g' = (I_c + q')h'(I_d + v').$$

It follows that

$$\begin{aligned} f + f' &= (I_a + p + I_c + p')(h + h')(I_b + u + I_d + u') \\ &= [I_{a+c} + (p + p')][(I_a + {}^{c}X^{y} + I_{y'})(h + h')(I_b + {}^{x}X^{d} + I_{x'})][I_{b+d} + (u + u')], \end{aligned}$$

and

$$g + g' = [I_{a+c} + (q + q')][(I_a + {}^{c}X^{y} + I_{y'})(h + h')(I_b + {}^{x}X^{d} + I_{x'})][I_{b+d} + (v + v')],$$

which imply $(f + f')\mathbf{R}(g + g')$.

Proposition 3.3 *The congruence $\mathbf{R}^+$ is the least initial and final congruence.*

Proposition 3.4 *If B is a $b\beta$-ssmc, then $B/\mathbf{R}$ is a strong $b\beta$-ssmc. If C is a strong $b\beta$-ssmc and $G : B \longrightarrow C$ is a $b\beta$-ssmc morphism, then there exists a unique $b\beta$-ssmc morphism $H : B/\mathbf{R} \longrightarrow C$ such that $G = F_B \bullet H$, where $F_B : B \longrightarrow B/\mathbf{R}$ is the canonical factorization morphism.*

The same factorisation gives us the left adjoints for the forgetful functors from the categories of $b\beta$-strong xy-ssmc to the categories of xy-ssmc, for every $x \in \{b, d\}$ and every $y \in \{\beta, \delta\}$. Since in any biflow, $\mathbf{R}$ is compatible with the feedback, the result holds also for biflows over the above xy-ssmc-ies.

4 The adjunction of $\perp$

We recall here briefly, following [4], the construction which associates to every ssmc a $b\alpha$-ssmc. We give some motivation. First, we mention the following identities in

a $b\alpha$-ssmc:

a) $f(\perp^x + I_b)g(\perp^y + I_c) = [f(I_x + g)](\perp^{x+y} + I_c)$
for $f : a \longrightarrow x + b$ and $g : b \longrightarrow y + c$.

b) $f(\perp^x + I_b) + g(\perp^y + I_d) = [(f + g)(I_x + {}^bX^y + I_d)](\perp^{x+y} + I_{b+d})$
for $f : a \longrightarrow x + b$ and $g : c \longrightarrow y + d$.

c) $f = f(\perp^0 + I_b)$ for $f : a \longrightarrow b$.

d) $\perp^x = I_x(\perp^x + I_0)$.

To obtain a $b\alpha$-ssmc from an ssmc B we have to add for every object x a new morphism $\perp^x : x \longrightarrow 0$. Consequently, it will be necessary to add for every morphism $f : a \longrightarrow x+b$ the morphism $f(\perp^x + I_b)$. The above identities imply that this suffices for our purpose. We represent the newly added morphism $f(\perp^x + I_b)$ as a pair $(f, x) : a \longrightarrow b$. Since in any ssmc $j\perp^x = \perp^y$ for every $a\alpha$-morphism $j : y \longrightarrow x$, we have $g(\perp^y + I_b) = g(j + I_b)(\perp^x + I_b)$ for every $g : a \longrightarrow y + b$. We deduce that the pairs (g, y) and $(g(j + I_b), x)$ represent the same morphism. Therefore we shall need a factorization which identifies them in order to accomplish our construction.

Let B be an ssmc with $(M, +, 0)$ the monoid of its objects. Consider category $K(B)$, having the same objects as B, with morphism defined for every $a, b \in M$ by

$$K(B)(a, b) := \{(f, x) | x \in M, f \in B(a, x + b)\}$$

and with composition defined by

$$(f, x)(g, y) := (f(I_x + g), x + y).$$

Note that the identity morphism of $a \in M$ in $K(B)$ is $(I_a, 0)$.

$K(B)$ becomes an snsmc defining the sum of $(f, x) : a \longrightarrow b$ and $(g, y) : c \longrightarrow d$ to be

$$(f, x) + (g, y) := ((f + g)(I_x + {}^bX^y + I_d), x + y)$$

and the distinguished morphism ${}^aX^b$ as $({}^aX^b, 0)$.

In $K(B)$ the distinguished morphisms $\perp^a := (I_a, a) \in K(B)(a, 0)$ have the following properties: $\perp^0 = I_0$ and $\perp^{a+b} = \perp^a + \perp^b$.

We define $I_B : B \longrightarrow K(B)$ as being the identity on objects and mapping every morphism f of B to $I_B(f) = (f, 0)$. I_B will be an M-snsmc morphism, and for every $(f, x) \in K(B)(a, b)$ we will have $(f, x) = I_B(f)(\perp^x + I_b)$, which is called the canonical decomposition of (f, x).

Definition 4.1 [4] *For every $a, b \in M$ we define a relation $\sim$ on $K(B)(a, b)$ in the following manner : $(f, x) \sim (g, y)$ in $K(B)(a, b)$ iff there exists an $a\alpha$-morphism $j \in B_a(y, x)$ such that $f = g(j + I_b)$.*

Note that $\sim$ is a congruence and that $K(B)/\sim$ is a $b\alpha$-ssmc.

The ssmc morphism $K_B : B \longrightarrow K(B)/\sim$ is by definition the composite of I_B with the canonical factorization morphism.

Proposition 4.2 *For every $b\alpha$-ssmc C and every ssmc morphism $F : B \longrightarrow C$ there exists a unique $b\alpha$-ssmc morphism $H : K(B)/\sim \longrightarrow C$ such that $F = K_B \bullet H$.*

Suppose now B is a biflow. Defining in $K(B)$ the right feedback of $(f,x) : a+c \longrightarrow b+c$ to be

$$(f,x)\uparrow^c = (f\uparrow^c, x)$$

we note that $K(B)$ becomes a flow. Because congruence $\sim$ is a flow congruence, $K(B)/\sim$ becomes a biflow over a $b\alpha$-ssmc and K_B becomes an M-biflow morphism. The next proposition is a version of Proposition 4.2.

Proposition 4.3 *For every biflow over a $b\alpha$-ssmc C and every biflow morphism $F : B \longrightarrow C$ there exists a unique biflow and $b\alpha$-ssmc morphism $H : K(B)/\sim \longrightarrow C$ such that $F = K_B \bullet H$.*

5 The strong adjunction of $\perp$

The results of Sections 2 and 4 imply that the passage from an ssmc B to a strong $b\alpha$-ssmc is a three step contruction: we construct first $K(B)$ and then we factor succesively through $\sim$ and $\mathbf{P}^+$. In this section we show that the two succesive factorsations can be replaced by a single one.

We give some motivation for the next definition. Since in any strong $b\alpha$-ssmc we have the identity $g(q+I_b)(\perp^x + I_b) = g(\perp^y + I_b)$ for every $g : a \longrightarrow y+b$ and $q : y \longrightarrow z$, we deduce that the pairs (g,y) and $(g(q+I_b),z)$ from $K(B)$ represent the same morphism. Analogously, for $f : a \longrightarrow x+b$ and $p : x \longrightarrow z$, the pairs (f,x) and $(f(p+I_b),z)$ represent the same morphism. Therefore, the equality $f(p+I_b) = g(q+I_b)$ is a sufficient condition to identify the pairs (f,x) and (g,y).

Definition 5.1 *For $(f,x),(g,y) \in K(B)(a,b)$ we say that $(f,x)\mathbf{Q}(g,y)$ iff there exist an object $z \in M$ and morphisms $p \in B(x,z)$ and $q \in B(y,z)$ such that $f(p+I_b) = g(q+I_b)$.*

Notice that $\mathbf{Q}$ is a reflexive and symmetric relation.

Lemma 5.2 *Relation $\mathbf{Q}$ is compatible with composition and sum.*

Proof. With the above notation, suppose $(f,x)\mathbf{Q}(g,y)$.

a) Compatibility with composition. Let $(f',x')\mathbf{Q}(g',y')$ in $K(B)(b,c)$, i.e. there exist an object $z' \in M$ and morphisms $p' \in B(x',z')$ and $q' \in B(y',z')$ such that $f'(p'+I_c) = g'(q'+I_c)$. We note that

$$\begin{aligned}[(f(I_x+f')][(p+p')+I_c] &= f(p+I_b)(I_x+f'(p'+I_c)) \\ &= g(q+I_b)(I_x+g'(q'+I_c)) \\ &= [g(I_y+g')][(q+q')+I_c],\end{aligned}$$

which implies that $[(f,x)(f',x')]\mathbf{Q}[(g,y)(g',y')]$.

b) Compatibility with sum. Suppose $(f',x')\mathbf{Q}(g',y')$ in $K(B)(c,d)$, i.e. there exist an object $z' \in M$ and morphisms $p' \in B(x',z')$ and $q' \in B(y',z')$ such that $f'(p'+I_d) = g'(q'+I_d)$. We note that

$$\begin{aligned}&[(f+f')(I_x + {}^bX^{x'} + I_d)][(p+p')+I_{b+d}] \\ &= [f(p+I_b) + f'(p'+I_d)](I_x + {}^bX^{x'} + I_d) \\ &= [[g(q+I_b) + g'(q'+I_d)](I_x + {}^bX^{x'} + I_d) \\ &= [(g+g')(I_y + {}^bX^{y'} + I_d)][(q+q')+I_{b+d}],\end{aligned}$$

which implies $[(f,x)+(f',x')]\mathbf{Q}[(g,y)+(g',y')]$.

It follows that the transitive closure $\mathbf{Q}^+$ of $\mathbf{Q}$ is a congruence. We denote by Q_B the composite of the morphism I_B, defined in Section 4, with the canonical factorization morphism $K(B) \longrightarrow K(B)/\mathbf{Q}$.

Proposition 5.3 *$K(B)/\mathbf{Q}$ is a strong $b\alpha$-ssmc. For every strong $b\alpha$-ssmc C and for every ssmc morphism $F : B \longrightarrow C$, there exists a unique $b\alpha$-ssmc morphism $H : K(B)/\mathbf{Q} \longrightarrow C$, such that $F = Q_B \bullet H$.*

Proof. We show first that $K(B)/\mathbf{Q}$ is an ssmc. For $(f,x) \in K(B)(a,b)$ and $(g,y) \in K(B)(c,d)$, taking into account that

$$[I_a + (g,y)][(f,x) + I_d] = ((f+g)({}^{x+b}X^y + I_d), y+x),$$

we note that there exists ${}^yX^x \in B(y+x, x+y)$ such that

$$(f+g)({}^{x+b}X^y + I_d)({}^yX^x + I_{b+d}) = (f+g)(I_x + {}^bX^y + I_d),$$

and thus $[I_a + (g,y)][(f,x) + I_d]\mathbf{Q}[(f,x)+(g,y)]$.

The existence of the distinguished morphisms $\perp^a = (I_a, a) \in K(B)(a,0)$ and their properties make $K(B)/\mathbf{Q}$ a $b\alpha$-ssmc. To prove that it is strong, let $(f,x) \in K(B)(a,0)$. The equality $f(I_x + I_0) = I_a(f + I_0)$ implies $(f,x)\mathbf{Q}\perp^a$.

Let $F : B \longrightarrow C$ be an ssmc morphism, with C a strong $b\alpha$-ssmc. We define $G : K(B) \longrightarrow C$ in the following way:

$$\begin{aligned} G(a) &:= F(a), \text{ for every object } a \in M, \\ G(f,x) &:= F(f)(\perp^{F(x)} + I_{F(b)}), \text{ for every } (f,x) \in K(B)(a,b). \end{aligned}$$

It follows that G is the unique snsmc morphism such that $F = I_B \bullet G$ and $G(\perp^a) = \perp^{G(a)}$ for every $a \in M$.

We now prove that $(f,x)\mathbf{Q}(g,y)$ in $K(B)(a,b)$ implies $G(f,x) = G(g,y)$. Let $p \in B(x,z)$ and $q \in B(y,z)$ be such that $f(p + I_b) = g(q + I_b)$. Applying F and then composing on the right with $\perp^{F(x)} + I_{F(b)}$ we deduce that

$$F(f)(\perp^{F(x)} + I_{F(b)}) = F(g)(\perp^{F(y)} + I_{F(b)}).$$

So, there exists $H : K(B)/\mathbf{Q} \longrightarrow C$ a unique $b\alpha$-ssmc morphism such that $F = Q_B \bullet H$.

Corollary 5.4 *$(K(B)/\sim)/\mathbf{P}$ is isomorphic to $K(B)/\mathbf{Q}$.*

Proposition 5.5 *If B is an ay-ssmc, where $y \in \{\alpha, \beta, \gamma, \delta\}$, then $K(B)/\mathbf{Q}$ is a $b\alpha$-strong by-ssmc and $Q_B : B \longrightarrow K(B)/\mathbf{Q}$ is an ay-ssmc morphism. If C is a $b\alpha$-strong by-ssmc and $F : B \longrightarrow C$ is an ay-ssmc morphism, then the unique $b\alpha$-ssmc morphism $H : K(B)/\mathbf{Q} \longrightarrow C$, such that $F = Q_B \bullet H$, given by Proposition 5.3, is a by-ssmc morphism.*

Proof. Case $y = \alpha$ is precisely Proposition 5.3. For the remainder of the cases, from the distinguished morphisms of B, $\top_a \in B(0,a)$ or $\vee_a \in B(a+a, a)$ we obtain the distinguished morphisms of $K(B)/\mathbf{Q}$, $Q_B(\top_a)$ or $Q_B(\vee_a)$.

The axioms fulfilled by $\top_a$ and/or $\vee_a$ in B, will be also fulfilled in $K(B)/\mathbf{Q}$. The only axioms which remain to be verified are those relating $\perp^a$ with $Q_B(\top_a)$ and $Q_B(\vee_a)$, that is

$$Q_B(\top_a)\perp^a = I_0 \text{ and } Q_B(\vee_a)\perp^a = \perp^a + \perp^a.$$

Their validity is a consequence of the fact that $K(B)/\mathbf{Q}$ is $b\alpha$-strong.

Proposition 5.6 *If B is a strong ay-ssmc, where $y \in \{\alpha, \beta, \gamma, \delta\}$, then $K(B)/\mathbf{Q}$ is a strong by-ssmc. If C is a strong by-ssmc and $F : B \longrightarrow C$ is an ay-ssmc morphism, then there exists a unique by-ssmc morphism $H : K(B)/\mathbf{Q} \longrightarrow C$ such that $F = Q_B \bullet H$.*

Proof. We prove only the first assertion. Assume $(f, x) \in K(B)(a, b)$.

For the distinguished morphism $\top_a$ of B, using the equalities

$$I_B(\top_a)(f, x) = (\top_a, 0)(f, x) = (\top_a(I_0 + f), x) = (\top_a f, x) = (\top_{x+b}, x)$$

and noting that

$$\top_{x+b}(I_x + I_b) = \top_b(\top_x + I_b),$$

it follows that $[I_B(\top_a)(f, x)]\mathbf{Q}I_B(\top_b)$.

For the distinguished morphisms $\vee_a$ of B, using the equalitites

$$I_B(\vee_a)(f, x) = (\vee_a f, x)$$

$$[(f, x) + (f, x)]I_B(\vee_b) = ((f + f)(I_x + {}^bX^x + I_b)(I_{x+x} + \vee_b), x + x)$$

and noting that

$$[(f + f)(I_x + {}^bX^x + I_b)(I_{x+x} + \vee_b)](\vee_x + I_b) = \vee_a f(I_x + I_b),$$

we deduce that $[((f, x) + (f, x))I_B(\vee_b)]\mathbf{Q}[I_B(\vee_a)(f, x)]$.

Suppose now that B is a biflow. Because the relation $\mathbf{Q}$ is compatible with the feedback, it follows that $K(B)/\mathbf{Q}$ is a biflow over a strong $b\alpha$-ssmc, and we obtain the corresponding version of Propositions 5.5 and 5.6.

Proposition 5.7 *If B is a biflow over an ay-ssmc, where $y \in \{\alpha, \beta, \gamma, \delta\}$, then $K(B)/\mathbf{Q}$ is a biflow over a $b\alpha$-strong by-ssmc and Q_B is a biflow and ay-ssmc morphism. If C is a biflow over a $b\alpha$-strong by-ssmc and $F : B \longrightarrow C$ is a biflow and ay-ssmc morphism, then there exists a unique biflow and by-ssmc morphism $H : K(B)/\mathbf{Q} \longrightarrow C$, such that $F = Q_B \bullet H$.*

Proposition 5.8 *If B is a biflow over a strong ay-ssmc, where $y \in \{\alpha, \beta, \gamma, \delta\}$, then $K(B)/\mathbf{Q}$ is a biflow over a strong by-ssmc. If C is a biflow over a strong by-ssmc and $F : B \longrightarrow C$ is a biflow and ay-ssmc morphism, then there exists a unique biflow and by-ssmc morphism $H : K(B)/\mathbf{Q} \longrightarrow C$ such that $F = Q_B \bullet H$.*

6 The simultaneous adjunction of $\top$ and $\bot$

In this section we show that the construction of Section 4, together with its dual, can be merged into a single one.

To understand the construction which follows we mention that for every $f : a + y \longrightarrow x + b$ the triple (y, f, x) represents the morphism $(I_a + \top_y)f(\bot^x + I_b)$, where $\bot^x : x \longrightarrow 0$ and $\top_y : 0 \longrightarrow y$ are the newly added morphisms. The definitions for composition and sum are based on the following identities which hold in any ssmc which is simultaneously an $a\beta$-ssmc and a $b\alpha$-ssmc:

$$\begin{aligned} a)\quad & (I_a + \top_y)f(\bot^x + I_b)(I_b + \top_z)g(\bot^w + I_c) \\ & = (I_a + \top_{y+z})[(f + I_z)(I_x + g)](\bot^{x+w} + I_c) \end{aligned}$$

where $f : a + y \longrightarrow x + b$ and $g : b + z \longrightarrow w + c$,

$$\begin{aligned} b)\quad & (I_a + \top_y)f(\bot^x + I_b) + (I_c + \top_z)g(\bot^w + I_d) \\ & = (I_{a+c} + \top_{y+z})[(I_a + {}^cX^y + I_z)(f + g)(I_x + {}^bX^w + I_d)](\bot^{x+w} + I_{b+d}) \end{aligned}$$

where $f : a + y \longrightarrow x + b$ and $g : c + z \longrightarrow w + d$.

Let B be an ssmc, with a fixed monoid of objects $(M, +, 0)$. Consider the category $J(B)$, having the same objects as B, and as morphisms, for each $a, b \in M$,

$$J(B)(a, b) := \{(y, f, x) | y, x \in M, f \in B(a + y, x + b)\}$$

with composition defined by

$$(y, f, x)(z, g, w) := (y + z, (f + I_z)(I_x + g), x + w).$$

Notice that the identiy morphism of $a \in M$ in $J(B)$ is $(0, I_a, 0)$, and it will be subsequently denoted by I_a.

In $J(B)$ define the sum of $(y, f, x) \in J(B)(a, b)$ and $(z, g, w) \in J(B)(c, d)$ as

$$(y, f, x) + (z, g, w) := (y + z, (I_a + {}^cX^y + I_z)(f + g)(I_x + {}^bX^w + I_d), x + w).$$

We prove next that $J(B)$ is an nsmc.

Let $(u, h, v) \in J(B)(p, q)$. We have

$$[(y, f, x) + (z, g, w)] + (u, h, v) =$$

$$= (y + z + u, (I_{a+c} + {}^pX^{y+z} + I_u)[(I_a + {}^cX^y + I_z)(f + g)(I_x + {}^bX^w + I_d) + h]\bullet$$

$$\bullet(I_{x+w} + {}^{b+d}X^v + I_q), x + w + v) =$$

$$= (y + z + u, (I_{a+c} + {}^pX^{y+z} + I_u)(I_a + {}^cX^y + I_{z+p+u})(f + g + h)\bullet$$

$$\bullet(I_x + {}^bX^w + I_{d+v+q})(I_{x+w} + {}^{b+d}X^v + I_q), x + w + v) =$$

$$= (y + z + u, (I_a + {}^{c+p}X^y + I_{z+u})(I_{a+y+c} + {}^pX^z + I_u)\bullet$$

$$\bullet(f + g + h)(I_{x+b+w} + {}^dX^v + I_q)(I_x + {}^bX^{w+v} + I_{d+q}), x + w + v) =$$

$$= (y + z + u, (I_a + {}^{c+p}X^y + I_{z+u})\bullet$$

$$\bullet[f + (I_c + {}^pX^z + I_u)(g + h)(I_w + {}^dX^v + I_q)](I_x + {}^bX^{w+v} + I_{d+q}), x + w + v) =$$

$$= (y, f, x) + [(z, g, w) + (u, h, v)].$$

It is easy to prove that $(y, f, x) + I_0 = (y, f, x) = I_0 + (y, f, x)$ and that $I_a + I_b = I_{a+b}$.

For Axiom 4a, we take $(y, f, x) : a \longrightarrow b, (z, g, w) : c \longrightarrow d$ and $(u, h, v) : d \longrightarrow e$, and we have

$$[(y, f, x) + (z, g, w)][I_b + (u, h, v)] =$$

$$= (y+z+u, [(I_a + {}^cX^y + I_z)(f+g)(I_x + {}^bX^w + I_d) + I_u][I_{x+w} + (I_b + h)({}^bX^v + I_e)], x+w+v) =$$

$$= (y + z + u, (I_a + {}^cX^y + I_{z+u})[f + (g + I_u)(I_w + h)](I_x + {}^bX^{w+v} + I_e), x + w + v) =$$

$$= (y, f, x) + (z, g, w)(u, h, v).$$

The proof of Axiom 4b is analogous.

Define now the distinguished symmetry morphisms of $J(B)$ to be ${}^aX^b := (0, {}^aX^b, 0)$. It can be proven that $J(B)$ is now a snsmc. We prove here only the validity of Axiom 5a.

$${}^aX^c(I_c + (y, f, x)){}^cX^b =$$

$$= (y, ({}^aX^c + I_y)(I_c + f)({}^cX^x + I_b)(I_x + {}^cX^b), x) =$$

$$= (y, ({}^aX^c + I_y)(I_c + f){}^cX^{x+b}, x) =$$

$$= (y, ({}^aX^c + I_y){}^cX^{a+y}(f + I_c), x) =$$

$$= (y, (I_a + {}^cX^y)(f + I_c), x) =$$

$$= (y, f, x) + I_c.$$

Define the canonical M-snsmc morphism $I_B : B \longrightarrow J(B)$ to be, for every $f \in B(a, b)$,

$$I_B(f) := (0, f, 0).$$

In $J(B)$ we define the distinguished morphisms

$$\perp^a := (0, I_a, a) \text{ and } \top_a := (a, I_a, 0)$$

and we notice that

$$\perp^0 = I_0 = \top_0$$

$$\perp^a + \perp^b = \perp^{a+b} \text{ and } \top_a + \top_b = \top_{a+b}.$$

For every $(y, f, x) \in J(B)(a, b)$ the following identity holds:

$$(y, f, x) = (I_a + \top_y) I_B(f)(\perp^x + I_b).$$

Definition 6.1 *For every $a, b \in M$, we define relation $\equiv$ in $J(B)(a, b)$ in the following manner: $(y, f, x) \equiv (z, g, w)$ iff there exist morphisms $k \in B_a(y, z)$ and $j \in B_a(w, x)$ such that $f = (I_a + k)g(j + I_b)$.*

Lemma 6.2 *The relation $\equiv$ is a congruence.*

Proof. Because every $a\alpha$-morphism is invertible, its inverse being also an $a\alpha$-morphism, it follows easily that $\equiv$ is an equivalence.

To prove compatibility with composition, using the same notation as in Definition 6.1, let $(y, f, x) \equiv (z, g, w)$ and let $(y', f', x') \equiv (z', g', w')$ in $J(B)(b, c)$, i.e. there exist $k' \in B_a(y', z')$ and $j' \in B_a(w', x')$ such that $f' = (I_b + k')g'(j' + I_c)$. We have

$$(f + I_{y'})(I_x + f') =$$

$$= (I_a + k + I_{y'})(g + I_{y'})(j + I_{b+y'})(I_{x+b} + k')(I_x + g')(I_x + j' + I_c) =$$

$$= [I_a + (k + k')][(g + I_{x'})(I_w + g')][(j + j') + I_c],$$

which implies that $(y, f, x)(y', f', x') \equiv (z, g, w)(z', g', w')$.

To prove compatibility with sum, let $(y, f, x) \equiv (z, g, w)$ again as in Definition 6.1, and $(y', f', x') \equiv (z', g', w')$ in $J(B)(c, d)$, i.e. there exist $k' \in B_a(y', z')$ and $j' \in B_a(w', x')$ such that $f' = (I_c + k')g'(j' + I_d)$. It follows that

$$(I_a + {}^cX^y + I_{y'})(f + f')(I_x + {}^bX^{x'} + I_d) =$$

$$= (I_a + {}^cX^y + I_{y'})(I_a + k + I_c + k')(g + g')(j + I_b + j' + I_d)(I_x + {}^bX^{x'} + I_d) =$$

$$= [I_{a+c} + (k + k')][(I_a + {}^cX^z + I_{x'})(g + g')(I_w + {}^bX^{w'} + I_d)][(j + j') + I_{b+d}],$$

which implies that $(y, f, x) + (y', f', x') \equiv (z, g, w) + (z', g', w')$.

Proposition 6.3 *$J(B)/\equiv$ is an $a\beta$-ssmc and a $b\alpha$-ssmc.*

Proof. We will prove only the permutability of $J(B)/\equiv$, that is, we will show, for every $(y, f, x) \in J(B)(a, b)$ and $(z, g, w) \in J(B)(c, d)$, that

$$(y, f, x) + (z, g, w) \equiv (I_a + (z, g, w))((y, f, x) + I_d).$$

This follows easily from the following calculations:

$$(I_a + (z, g, w))((y, f, x) + I_d) =$$

$$= (z, (I_a + g)({}^aX^w + I_d), w) \bullet (y, (I_a + {}^dX^y)(f + I_d), x) =$$

$$= (z + y, (I_a + g + I_y)({}^aX^w + I_{d+y})(I_{w+a} + {}^dX^y)(I_w + f + I_d), w + x) =$$

$$= (z + y, (I_a + g + I_y)(I_a + {}^{w+d}X^y)({}^{a+y}X^w + I_d)(I_w + f + I_d), w + x) =$$

$$= (z + y, (I_a + {}^{c+z}X^y)(I_{a+y} + g)(f + I_{w+d})({}^{x+b}X^w + I_d), w + x) =$$

$$= (x + y, (I_{a+c} + {}^zX^y)[(I_a + {}^cX^y + I_z)(f + g)(I_x + {}^bX^w + I_d)]({}^xX^w + I_{b+d}), w + x).$$

Let $J_B : B \longrightarrow J(B)/\equiv$ be the ssmc morphism obtained by compsoing I_B with the canonical factorization morphism.

Proposition 6.4 *If C is an $a\beta$-ssmc and a $b\alpha$-ssmc and $F : B \longrightarrow C$ is an ssmc morphism, then there exists a unique $a\beta$-ssmc and $b\alpha$-ssmc morphism $H : J(B)/\equiv \longrightarrow C$ such that $F = J_B \bullet H$.*

Proof. We will prove that there exists a unique snsmc morphism $G : J(B) \longrightarrow C$ such that $F = I_B \bullet G, G(\bot^a) = \bot^{G(a)}$ and $G(\top_a) = \top_{G(a)}$. Define G as follows:

$$G(a) := F(a), \text{ for every object } a \in M,$$

$$G(y, f, x) := (I_{F(a)} + \top_{F(y)}) \bullet F(f) \bullet (\bot^{F(x)} + I_{F(b)},) \text{ for every } (y, f, x) \in J(B)(a, b).$$

For $f \in B(a, b)$, notice that

$$G(I_B(f)) = G(0, f, 0) = F(f),$$

from which follows also that

$$G(I_a) = I_{G(a)} \text{ and } G({}^aX^b) = {}^{G(a)}X^{G(b)}.$$

For $(y, f, x) \in J(B)(a, b)$ and $(z, g, w) \in J(B)(b, c)$ we have:

$$G((y, f, x)(z, g, w)) = (I_{F(a)} + \top_{F(y+z)})F((f + I_z)(I_x + g))(\bot^{F(x+w)} + I_{F(c)}) =$$

$$= (I_{F(a)} + \top_{F(y)})(F(f) + \top_{F(z)})(\bot^{F(z)} + F(g))(\bot^{F(w)} + I_{F(c)}) =$$

$$= (I_{F(a)} + \top_{F(y)})F(f)(\bot^{F(x)} + I_{F(b)})(I_{F(b)} + \top_{F(z)})F(g)(\bot^{F(w)} + I_{F(c)}) =$$

$$= G(y, f, x) \bullet G(z, g, w).$$

For $(y, f, x) \in J(B)(a, b)$ and $(z, g, w) \in J(B)(c, d)$ we have:

$$G((y, f, x) + (z, g, w)) =$$

$$= (I_{F(a+c)} + \top_{F(y+z)})F((I_a + {}^cX^y + I_z)(f + g)(I_x + {}^bX^w + I_d))(\bot^{F(x+w)} + I_{F(b+d)}) =$$

$$= (I_{F(a)} + \top_{F(y)} + I_{F(c)} + \top_{F(z)})F(f + g)(\bot^{F(x)} + I_{F(b)} + \bot^{F(w)} + I_{F(d)}) =$$

$$= G(y, f, x) + G(z, g, w).$$

We also note that

$$G(\bot^a) = G(0, I_a, a) = (I_{F(a)} + \top_{F(0)})F(I_a)(\bot^{F(a)} + I_{F(0)}) = \bot^{G(a)}$$

and, similarly,

$$G(\top_a) = G(a, I_a, 0) = (I_{F(0)} + \top_{F(a)})F(I_a)(\bot^{F(0)} + I_{F(a)}) = \top_{G(a)}.$$

If $(y, f, x) \equiv (z, g, w)$, using the same notations as in Definition 6.1, we have

$$G(y, f, x) = (I_{F(a)} + \top_{F(y)})F(f)(\bot^{F(x)} + I_{F(b)}) =$$

$$= (I_{F(a)} + \top_{F(y)})(I_{F(a)} + F(k))F(g)(F(j) + I_{F(b)})(\bot^{F(x)} + I_{F(b)}) =$$

$$= (I_{F(a)} + \top_{F(z)})F(g)(\bot^{F(w)} + I_{F(b)}) = G(z, g, w),$$

from which we deduce the existence of the morphism H with the required properties.

Suppose for the rest of this section that B is a biflow.

For $(y, f, x) \in J(B)(c+a, c+b)$ we define the left feedback

$$\uparrow^c (y, f, x) := (y, \uparrow^c [f({}^x X^c + I_b)], x)$$

and we prove that $J(B)$ becomes a flow.

For $(z, g, w) : d \longrightarrow a$, we have:

$$\uparrow^c [(I_c + (z, g, w))(y, f, x)] =$$

$$= \uparrow^c (z + y, (I_c + g + I_y)({}^c X^w + I_{a+y})(I_w + f), w + x) =$$

$$= (z + y, \uparrow^c [(I_c + g + I_y)({}^c X^w + I_{a+y})(I_w + f)({}^{w+x} X^c + I_b)], w + x) =$$

$$= (z + y, (g + I_y)(I_w + \uparrow^c (f({}^x X^c + I_b))), w + x) =$$

$$= (z, g, w) \bullet \uparrow^c (y, f, x).$$

For $(z, g, w) : b \longrightarrow d$ we have

$$\uparrow^c [(y, f, x)(I_c + (z, g, w))] =$$

$$= \uparrow^c [(y, f, x)(z, (I_c + g)({}^c X^w + I_d), w)] =$$

$$= \uparrow^c (y + z, (f + I_x)(I_{x+c} + g)(I_x + {}^c X^w + I_d), x + w) =$$

$$= (y + z, \uparrow^c [(f + I_x)(I_{x+c} + g)({}^x X^c + I_{w+d})], x + w) =$$

$$= (y + z, \uparrow^c [(f({}^x X^c + I_b) + I_x)(I_c + I_x + g)], x + w) =$$

$$= (y + z, (\uparrow^c [f({}^x X^c + I_b) + I_x])(I_x + g), x + w) =$$

$$= (y + z, (\uparrow^c (f({}^x X^c + I_b)) + I_x)(I_x + g), x + w) =$$

$$= \uparrow^c (y, f, x) \bullet (z, g, w).$$

Furthermore,

$$\uparrow^c [(y, f, x) + I_d] = \uparrow^c (y, (I_{c+a} + {}^d X^y)(f + I_d), x) =$$

$$= (y, \uparrow^c [(I_{c+a} + {}^d X^y)(f + I_d)({}^x X^c + I_{b+d}], x) =$$

$$= (y, (I_a + {}^d X^y)[\uparrow^c (f({}^x X^c + I_b)) + I_d], x) = \uparrow^c (y, f, x) + I_d.$$

For $(y, f, x) : c + d + a \longrightarrow c + d + b$ we have:

$$\uparrow^{d+c} [({}^d X^c + I_a)(y, f, x)({}^c X^d + I_b)] =$$

$$= \uparrow^{d+c} (y, ({}^d X^c + I_{a+y}) f(I_x + {}^c X^d + I_b), x) =$$

$$= (y, \uparrow^{d+c} [({}^d X^c + I_{a+y}) f(I_x + {}^c X^d + I_b)({}^x X^{d+c} + I_b)], x) =$$

$$= (y, \uparrow^{d+c} [({}^d X^c + I_{a+y}) f({}^x X^{c+d} + I_b)({}^c X^d + I_{x+b})], x) =$$

$$= (y, \uparrow^{c+d} [f({}^x X^{c+d} + I_b)], x) = \uparrow^{c+d} (y, f, x).$$

Noticing that for $f \in B(c+a, c+b)$ we have

$$\uparrow^c I_B(f) = \uparrow^c (0, f, 0) = (0, \uparrow^c f, 0) = I_B(\uparrow^c f),$$

it follows easily that $\uparrow^c I_c = I_0$ and $\uparrow^c {}^c X^c = I_c$.

We prove now that $\equiv$ is a flow congruence. Suppose $(y,f,x) \equiv (z,g,w)$ in $J(B)(c+a,c+b)$, i.e. there exist $a\alpha$-morphisms $k \in B_a(y,z)$ and $j \in B_a(w,x)$ such that $f = (I_{c+a}+k)g(j+I_{c+b})$, and we notice that

$$\uparrow^c (y,f,x) = (y, \uparrow^c [(I_{c+a}+k)g(j+I_{c+b})({}^xX^c + I_b)], x) =$$

$$= (y, (I_a+k)(\uparrow^c [g({}^wX^c + I_b)])(j+I_b), x) \equiv \uparrow^c (z,g,w).$$

It follows that $J(B)/\equiv$ is a biflow and $J_B : B \longrightarrow J(B)/\equiv$ is a biflow morphism.

Proposition 6.5 *If C is a biflow over an $a\beta$-ssmc and a $b\alpha$-ssmc and if $F : B \longrightarrow C$ is a biflow morphism, then there exists a unique biflow, $a\beta$-ssmc and $b\alpha$-ssmc morphism $H : J(B)/\equiv \longrightarrow C$, such that $F = J_B \bullet H$.*

Proof. Using the proof of proposition 6.4 and keeping the same notation, it is sufficient to prove that G is a flow morphism.

Let $(y,f,x) \in J(B)(c+a,c+b)$. Then we have:

$$\uparrow^{G(c)} G(y,f,x) = \uparrow^{F(c)} [(I_{F(c+a)} + \top_{F(y)})F(f)(\bot^{F(x)} + I_{F(c+b)})] =$$

$$= (I_{F(a)} + \top_{F(y)}) \uparrow^{F(c)} [F(f)({}^{F(x)}X^{F(c)} + I_{F(b)})](\bot^{F(x)} + I_{F(b)}) =$$

$$= (I_{F(a)} + \top_{F(y)})F(\uparrow^c [f({}^xX^c + I_b)])(\bot^{F(x)} + I_{F(b)}) =$$

$$= G(\uparrow^c (y,f,x)).$$

References

[1] S.L. Bloom, Z. Ésik: Iteration algebras, Technical Report 9009, Stevens Institute of Technology.

[2] S.L. Bloom, R. Tindell: A note on zero-congruences, Acta Cybernetica, 8 (1987), 1-4.

[3] V. E. Căzănescu, Gh. Stefănescu: Towards a new algebraic foundation of flowchart scheme theory, Preprint Series in Mathematics No. 43/1987, INCREST Bucharest; Fundamenta Informaticae XIII (1990), 171-210.

[4] V. E. Căzănescu, Gh. Stefănescu: Classes of finite relations as initial abstract data types, Preprint Series in Mathemtics Nos. 3, 34 and 47/1989, INCREST Bucharest; First part in Discrete Mathematics 90 (1991), 233-265.

[5] V.E. Căzănescu, Gh. Stefănescu: A formal representation of flowchart schemes, An. Univ. Bucuresti, Mat.-Inf. 37 (1988), 33-51.

[6] V.E. Căzănescu, Gh. Stefănescu: A formal represenation of flowchart schemes II, Stud. Cerc. Mat. 41 (1989), 3, 151-167.

[7] V.E. Căzănescu, Gh. Stefănescu: A general result on abstract flowchart schemes with applications to the study of accessiblity, reduction and minimization, Preprint Series in Mathematics No. 7/1990, INCREST Bucharest; Theoret. Comput. Sci. 99 (1992), 1-63.

Received April 11, 1991.

Acta Cybernetica, Vol. 10, Nr. 3, Szeged, 1992

A note on intersections of isotone clones

János Demetrovics and Lajos Rónyai *

Abstract

We show that for every $k > 3$ there exists two chains P_1, P_2 over a base set A, $|A| = k$ such that the only isotone functions P_1 and P_2 have in common are the constants and projections. This settles a question raised by Demetrovics, Miyakawa, Rosenberg, Simovici and Stojmenović. We prove a related result which generalises the observation that two 3-element chains over the same ground set always admit a nontrivial common order preserving operation.

1 Introduction

Let A be a nonempty finite set. An *n-ary operation over A* is a function from A^n to A. $O_n(A)$ denotes the set of all n-ary operations over A and we put $O(A) = \cup_{n\geq 0} O_n(A)$. A set of operations $C \subseteq O(A)$ is a *clone* over A if it contains the projections and is closed under arbitrary superpositions (cf. Jablonskii [J58], Pöschel, Kaluznin [PK79], Szendrei [SZ86]). The set of all clones over A is denoted by $L(A)$. $L(A)$ is a partially ordered set with respect to inclusion and is closed under intersection. Clearly the set K_A of all projections and constant operations form a clone over A.

Let $P =< A, \leq >$ be a partial order (a poset for short) on A. We say that an operation $f \in O_n(A)$ *preserves* P if $x_1 \leq y_1$, $x_2 \leq y_2, \ldots, x_n \leq y_n$ implies that $f(x_1, x_2, \ldots, x_n) \leq f(y_1, y_2, \ldots, y_n)$, for every $x_i, y_i \in A$. In this case f is called an *isotone function* (with respect to P). It is easy to see that

$$Pol(P) = \{f \in O(A);\ f \text{ preserves } P\}$$

is a clone over A and $Pol(P) \supseteq K_A$. In [DMRSS90] Demetrovics, Miyakawa, Rosenberg, Simovici and Stojmenović studied intersections of clones of the form $Pol(P)$. In the context of semirigid relations they proved that if $|A| > 7$ or $|A| = 6$ then there exists two posets P_1, P_2 over A for which we have $Pol(P_1) \cap Pol(P_2) = K_A$. Also, they constructed four chains Q_1, Q_2, Q_3, Q_4 over A for which the clones $Pol(Q_i)$ intersect in K_A. The objective of this note is to improve the latter result. For $|A| > 3$ we exhibit two chains P_1, P_2 over A with the property $Pol(P_1) \cap Pol(P_2) = K_A$ (Theorem A). It is easy to see that any two chains over a 3-element set admit a common order preserving function. This observation is generalised in Theorem B. We show for a large class of posets P that any two isomorphic copies of P over the same ground set have a common order preserving operation. This class, besides the 3-element chain, includes the diamond and the pentagon. The note is concluded with a problem for further research.

*Computer and Automation Institute, Hungarian Academy of Sciences Budapest, Victor Hugo u. 18-22. H-1132 Hungary. Research partially supported by OTKA Grant 2581.

2 The results

Recall that a pair of elements $a < b$ of a poset P forms a *cover* if there is no $c \in P$ such that $a < c < b$. In this case we say also that b is an *upper cover* of a and a is a *lower cover* of b. A poset P is *bounded* if there exist $x, y \in P$ such that for every $z \in P$ we have $x \leq z \leq y$. In the sequel we shall use the following result (cf. [LP84], [P84]).

Lemma 1 *Let $|A| > 2$ and C be a clone over A. Then $C = K_A$ if and only if $C \cap O_1(A) = K_A \cap O_1(A)$.*

In simple terms, Lemma 1 states that a clone C is K_A exactly when the unary fuctions in C are the constants and the identity function. For $k > 0$ let A_k denote the set $\{0, 1, \ldots k-1\}$.

Theorem A. *For every integer k Ú 3 there exists two chains P_1, P_2 on A_k such that $Pol(P_1) \cap Pol(P_2) = K_{A_k}$.*

Proof. We give first the definitions of P_1 and P_2 by specifying the covers in the respective orders:

$$P_1: \quad 0 < 1 < 2 < \ldots < k-2 < k-1.$$

In the definition of P_2, we distinguish two cases, corresponding to the parity of k. If $k = 2m$ then we put

$$P_2: \quad 2m-2 < 2m-4 < \ldots < 2 < 0 < 2m-1 < 2m-3 < \ldots < 3 < 1.$$

If $k = 2m+1$ then we set

$$P_2: \quad 2m-2 < 2m-4 < \ldots < 2 < 2m < 0 < 2m-1 < 2m-3 < \ldots < 3 < 1.$$

In other words, P_1 is the standard ordering of A_k, while in P_2 we have first the *even* integers from the interval $[0, k-1]$ in a decreasing order (with respect to the standard ordering) followed by the odd numbers listed decreasingly again, provided that k is even. If k is odd then a little perturbation is introduced: $k-1$ is placed between 2 and 0 rather than to the beginning of the sequence. This is possible because $k > 3$ and therefore $2 \neq 2m$.

As for the proof, let $f \in Pol(P_1) \cap Pol(P_2)$ be a nontrivial unary function (i.e. f is not constant and not the identical function on A_k). Chains have no nontrivial automorphisms, therefore there exists $a \neq b \in A_k$ such that $f(a) = f(b)$. Using that $f \in Pol(P_1)$, we can assume that $b = a+1$, hence a and b have different parities. Now from $f \in Pol(P_2)$ we infer that $f(0) = f(k-1)$ if k is even and $f(0) = f(k-2)$ if k is odd. Switching back to P_1 we obtain that $f(0) = f(1) = \cdots = f(k-1)$ for k even. In this case the proof is finished. For k odd the same argument gives that $f(0) = f(1) = f(2) = \cdots = f(k-2)$. From the relations $2 < 2m < 0$ in P_2 we infer $f(0) = f(2m) = f(2)$ and conclude that f is a constant. The proof is complete. □

The unary functions over A_2 are the identity function and the constants. If P_1 and P_2 are chains over A_3 then an easy argument shows that $Pol(P_1) \cap Pol(P_2)$ is nontrivial. Next we prove a generalization of this observation. A finite bounded poset has the *cover property* if every element, except possibly the least and the greatest elements, has either a unique lower cover or a unique upper cover. We argue that there are many posets having the cover property. In fact, if P is an arbitrary

bounded poset then if we replace every $z \in P$ (except possibliy the greatest and the least elements of P) by a two-element chain then the resulting poset will have the cover property.

Theorem B. *Let P be a bounded poset on the finite base set A. Let $0, 1 \in A$ denote the least and the greatest elements of P. Suppose that there is an element $a \in P$ such that $0 < a$ and $a < 1$ are covers and that the poset $P \setminus \{a\}$ has the cover property. Let Q be an other poset on the base set A isomorphic to P. Then $Pol(P)$ and $Pol(Q)$ have a nontrivial intersection, i.e. $Pol(P_1) \cap Pol(P_2) \supset K_A$.*

Proof. Let $\phi : A \to A$ denote the map establishing an isomorphism $phi : P \to Q$ and put $b = \phi(a)$. Observe first that an arbitary map $f : P \to P$ which is the identical map on $P \setminus \{a\}$ is actually an order preserving map of P. For this reason if $b = a$ then for the map $g : A \to A$ defined as $g(a) = 1$ and $g(y) = y$ if $y \neq a$ we have $g \in Pol(P) \cap Pol(Q)$. We can henceforth assume that $a \neq b$. If $b \notin \{0, 1\}$ then we can easily construct a nontivial function $h \in Pol(P) \cap Pol(Q)$ as follows. As $P \setminus \{a\}$ has the cover property, $b \in P$ has either a unique upper cover in P or a unique lower cover in P. We shall assume that $c \in P$ is a unique upper cover of b in P (the other case can be treated in exactly the same way). Now set $h(b) = c$ and $h(z) = z$ if $z \in A \setminus \{b\}$. From the fact that c is a unique upper cover of b in $P \setminus \{a\}$ an therefore in P, we obtain that $h \in Pol(P)$. By our first observation we have $h \in Pol(Q)$ as well.

We are left with four cases to consider: $a \neq b$, $b \in \{0, 1\}$ and (by symmetry) $a \in \{\phi(0), \phi(1)\}$. In each case we shall define a nontrivial unary function $h \in Pol(P) \cap Pol(Q)$.

(i) If $b = 0$ and $a = \phi(0)$ then we set $h(a) = h(b) = a$ and $h(y) = 1$ if $y \notin \{a, b\}$.
(ii) Analogously, if $b = 1$ and $a = \phi(1)$ then we set $h(a) = h(b) = a$ and $h(y) = 0$ if $y \notin \{a, b\}$.
(iii) If $b = 0$ and $a = \phi(1)$ then we set $h(a) = h(b) = a$ and $h(y) = 1$ if $y \notin \{a, b\}$.
(iv) Analogously, if $b = 1$ and $a = \phi(0)$ then we put $h(a) = h(b) = a$ and $h(y) = 0$ if $y \notin \{a, b\}$.

In all cases we have $|Im(h)| = 2$ therefore h neither is constant nor is the identity function on A. The easy verification of the fact that h is an isotone function with respect to both P and Q is left to the reader.

□

Corollary C. *Let P and Q be two posets on A_5 isomorphic to the pentagon (i.e. the poset on A_5 defined by the covers $0 < 1 < 2 < 3$ and $0 < 4 < 3$). Then $Pol(P)$ and $Pol(Q)$ have a nontrivial intersection.*

□

Example. In contrast to Corollary C, consider the posets R and S over the base set A_6 defined by covers as follows:

$$R: \quad 0 < 1 < 2 < 3 \text{ and } 0 < 4 < 5 < 3.$$

$$S: \quad 1 < 3 < 0 < 5 \text{ and } 1 < 4 < 2 < 5.$$

Note that R is obtained from the pentagon by inserting a new element between 4 and 3. Clearly R and S are isomorphic posets. We show that $Pol(R)$ and $Pol(S)$ have a trivial intersection, i.e. $Pol(R) \cap Pol(S) = K_{A_6}$.

To this end, let $f \in Pol(R) \cap Pol(S)$ be a unary function. We consider first the case when $f(0) \neq 0$ or $f(3) \neq 3$. We claim that in this case $|Im(f)| \leq 2$. Indeed, $f \in Pol(R)$ implies then that $Im(f)$ is bounded in R and is consequently a subset

of one of the following four sets: $\{0,1,2\}$, $\{0,4,5\}$, $\{1,2,3\}$ and $\{3,4,5\}$. On the other hand, $Im(f)$ is a bounded poset with respect to S as well. As neither of the above four subsets of A_6 form a bounded subposet of S, the claim follows. If f is not a constant then we have $|Im(f)| = 2$ and $f(0) \not/ f(3)$. Now an inspection of S reveals that $f(1) = f(3)$ and $f(5) = f(0)$. Using again that $f \in Pol(R)$ we obtain that $f(2) = f(3)$ and $f(4) = f(5)$. The latter implies in S that $f(2) = f(5)$, showing that f is a constant, a contradiction.

From now on we can assume that $f(0) = 0$ and $f(3) = 3$. Now $f \in Pol(S)$ implies that $f(5) \in \{0,5\}$ and $f(1) \in \{1,3\}$. But $f(5) = 0$ would imply in R that $f(4) = 0$ which in S leads to $f(2) = 0$. The latter in R implies $f(1) = 0$ which in S leads to the contradictory $f(3) = 0$. A similar argument switching back and forth between R and S shows that $f(1) = 1$. At this point we have $f(i) = i$ for $i \in \{0,1,3,5\}$ and (from R) $f(4) \in \{0,4,5\}$. Here $f(4) \in \{0,5\}$ would give (in S) that $f(2) \in \{0,5\}$, which contradicts the relation

$$(*) \qquad f(2) \in \{1,2,3\}$$

obtained from R. We infer that $f(4) = 4$ and this gives in S that $f(2) \in \{2,4,5\}$. This together with (*) implies that $f(2) = 2$, i.e. f is the identity function of A_6. This proves the statement.

Motivated by our considerations we propose the follwing open research problem.

Problem. Find a characterization of the (bounded) posets $P =< A, \leq_P >$ for which there exists a poset $Q =< A, \leq_Q >$ such that P and Q are isomorphic and $Pol(P) \cap Pol(Q) = K_A$.

References

[DMRSS90] J. Demetrovics, M. Miyakawa, I. G. Rosenberg, D. A. Simovici, I. Stojmenović: Intersections of isotone clones on a finite set; *Proc. of the 20th International Symposium on Multiple-Valued Logic, Charlotte, NC, 1990, 248-253.*

[J58] S. V. Jablonskii: Functional constructions in a k-valued logic (Russian); *Trudy Mat. Inst. Steklov, 51 (1958) 5-142.*

[LP84] F. Länger, R. Pöschel: Relational systems with trivial endomorphisms and polymorphisms; *J. Pure and Appl. Algebra 32 (1984) 129-142.*

[P84] P. P. Pálfy: Unary polynomials in algebras I; *Algebra Universalis 18 (1984) 262-273.*

[PK79] R. Pöschel, L. A. Kaluznin: Funktionen und relationenalgebren; *VEB Deutscher Verlag der Wissenschaften, Berlin 1979.*

[SZ86] Á. Szendrei: Clones in universal algebra; *Les Presses de l'Université de Montréal, 1986.*

Received September 2, 1991.

Készítette a JATEPress
6722 Szeged, Petőfi Sándor sugárút 30—34.

Felelős szerkesztő: Szőnyi Etelka
Sokszorosító vezető: Szögi Lászlóné
Méret: B/5, példányszám: 400, munkaszám: 240/92.

Subscription information and mailing address for editorial correspondence:

Acta Cybernetica
Árpád tér 2.
Szeged
H-6720 Hungary

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1993. január
Terjedelem: 7,12 (B/5) iv